The Path Scheduling Problem

Consider a network and a set of paths with congestion C and dilation D. Initial queues are each of size 1, but the final queues can consume all packets.

We want to show that for a leveled network, we can solve this in $O(C + D + \log N)$ time whp. An algorithm that solves this problem is Ranade's Algorithm, which is an on-line algorithm.

Theorem 1: $\exists$ *a schedule such that*

(1) *each edge is used once in a time step*

(2) *queues are of constant size*

(3) *length of schedule is within a constant of the best possible ($O(C + D)$)*

We don't know how, but it can be done. Off-line, it can be done in polynomial time. It can be done on-line with some loss in performance.

Applications:

(1) Simulate a graph G on a host network H in $O(C + D)$ time if G can be imbedded into H with congestion C and dilation D.

(2) Have a set of jobs $J = \{j_1, \ldots, j_n\}$ and machines $M = \{m_1, \ldots, m_k\}$. Each job runs for one unit of time on any machine and each job runs only once per machine. For a lower bound on the run time:

Let $c =$ max number of jobs that use any machine and $d =$ max job length. The following is almost a proof of Theorem 1.

Theorem 2: $\exists$ *a randomized algorithm that produces a schedule of length $O(c + d + \log Nd)$.*

That algorithm is: each packet picks a random δ from $[0, \frac{\alpha c}{\log Nd}]$. Wait δ and then move greedily to destination. This is not that great, since it's not a real schedule. The "schedule" this algorithm gives you allows many packets to travel along the same edge at the same time. To convert it to a real schedule, replace each time step with C_{greedy} time steps, where C_{greedy} is the max number of packets that want to use any edge in one time step of the fake schedule. If n packets need to use the same edge in a time step of the fake schedule, move them and everything else along *exactly one vertex* in n time steps in the real schedule. The time for the real schedule is $(d + \frac{\alpha c}{\log Nd})C_{greedy}$.

Claim: $C_{greedy} = \log dN$ whp.

$Pr[k$ packets use edge e at time t in greedy random delay schedule$] \le \binom{c}{k} (\frac{\log Nd}{\alpha c})^k \le (\frac{ce}{k})^k (\frac{\log Nd}{\alpha c})^k$.

If we let $\alpha = 4e$, then $(\frac{e \log Nd}{\alpha k})^k \le (\frac{1}{4})^{\log Nd} \le (\frac{1}{\log Nd})^2$, so $Pr[$any edge is bad$] \le (Nd)(\frac{1}{Nd})^2$. **Note:** this theorem won't work for constant-sized queues. Sorry. You need queues of length $\log Nd(C_{greedy})$.

Theorem 3: $\exists$ *a schedule of length* $(c + d)^{O(\log^* n)}$ *with queues of length* $(\log d)2^{O(\log^* n)}$.

Lovasz Local Lemma (LLL): let $A_1, \ldots, A_m$ be a set of bad events, each occuring with a probability $\le p$ and is independent of all but b other variables. If $4bp < 1$, then with probability > 0, no bad event occurs.

Definition: a T-frame is a set of consecutive T-steps. The frame congestion, C, of a T-frame is the maximum number of packetsthat use any edge of the T-frame. Relative frame congestion is $\frac{C}{T}$.

Lemma: $\exists$ *a schedule of length $O(C + D)$ in which packets only wait at edge queues and relative congestion is 1 on T-frames with $T \ge \log d$.*

Proof: let $C = d$. Assign a random delay $\delta \in [0, \frac{\alpha c}{\log Nd}]$. Wait δ time steps and then go greedily. $A_{e,t,T}$ is the bad event that edge e has relative congestion > 1 for a T-frame starting at t $\forall T > \log d$. This event depends on event $A_{e',t',T'}$ for at most Cd choices of e' and $(1 + \alpha)d$ choices of t' and T', so (from LLL), $b \le d^4(1 + \alpha^2)$. $Pr[A_{e,t,T}] \le \binom{C}{T} (\frac{T}{\alpha d})^T \le (\frac{ce}{T})^T (\frac{T}{\alpha d})^T \le (\frac{e}{\alpha})^T$. If we let $\alpha = 32e$, we find the probability is $< (\frac{1}{32})^T$. If we let $T = \log d$, the probability is $\le (\frac{1}{2^5})^{\log d} \le \frac{1}{d^5}$, so $4bp \le \frac{4(1+\alpha^2)}{d} \le 1$.