In this lecture we talk about extensions of the simple admission control algorithm from last class. We will talk about 3 extensions:

1. Admission control if there is a limit L on the maximum number of edges allowed in any path.
2. Admission control for multicast.
3. Admission control if the requests demand different bandwidth, and the edges have different bandwidth.

Admission control for short path. The **problem** we consider is as follows:

- Given a graph G with n nodes and m edges (undirected graph), a bound on path length L, and a bound $k \approx \log_2 m$.
- pairs of nodes arrive on-line
- When a pair (s,t) arrives we either reject the pair, or accept it, and route it in the graph on a path with at most L edges so that on any edge we may route at most k paths.
- The objective is to maximize the number of paths accepted.

The algorithm is almost the same as before. The only change needed is to use $cost_L(s,t)$ instead of $cost(s,t)$, where $cost_L(s,t)$ is the minimum cost of an path from s to t that uses at most L edges.

First we show that $cost_L(s,t)$ and the corresponding path can be computed efficiently.

Lemma 1 *Given two nodes s and t the $cost_L(s,t)$ and the corresponding path can be computed in $O(mL)$ time, where m is the number of edges in the graph.*

Proof. We use a recursion also used in the Bellman-Ford algorithm, and compute $cost_i(s,v)$ for all node v and all $i = 1,\ldots,L$. Note that $cost_0(s,v)$ is the costs of the edge (s,v) or ∞ if no such edge exists. To compute $cost_i(s,v)$ we use the formula:

$$cost_i(s,v) = \min(cost_{i-1}(s,v), \min_{w\in V} cost_{i-1}(s,w) + c(w,v)),$$

as the min-cost path using at most i edges either has only $i-1$ edges, or it consists of a path of $i-1$ edges to some other node w followed by the edge (w,v).

Given the costs $cost_{i-1}(s,v)$ for all nodes v and a value i we can compute all the costs $cost_i(s,v)$ in $O(m)$ time: one update for every edge (w,v). So the time to compute all $cost_i(s,v)$ values is $O(mL)$. ■

Next we observe that the proof last time extends step by step without any change (by replacing $cost(s,t)$ by $cost_L(s,t)$ every time). We get the theorem as before.

Theorem 1 *If $k \geq \log_2 m$, $k \leq O(\log m)$ and $\mathcal{O}_L$ and $\mathcal{A}_L$ is number of path accepted by an optimum solution and the algorithm, then $|\mathcal{O}_L| \leq (2k+1)|\mathcal{A}_L|$.*

In the homework we will improve this algorithm to get an algorithm with $k = \theta(\log L)$ instead.

Admission control for multicast Next we consider the version when the connection requests are for *sets* of nodes to be connected instead of pairs. This is known as the multicast admission control problem. Each group, if accepted, will be connected by a tree, and we have to make sure that no more than k trees use the same edge. Note that when a member of a group wants to sent information to all other members, it will send the information along the tree. Now if a tree node has degree 3 or higher, than this node will have to forward the information to more than one node, i.e., will have to copy the information.

The idea of the algorithm is same as before: we introduce $f(e)$ to denote the number of trees through edge e, and use the cost function $c(e) = 2^{f(e)}$. For a connection request for a set S, we will want to compute $cost(S)$, the minimum cost tree that contains S. Note that this is potentially different from the minimum spanning tree on S, as the minimum cots tree might contain nodes other than S. The problem of computing a minimum cost tree that contains a set S is called the *minimum cost Steiner Tree* problem. A problem with this approach is that the Steiner Tree problem is NP-hard. Next class we will show that there is a 2-approximation algorithm for the problem.

Theorem 2 *There is a polynomial time algorithm that given a Steiner Tree problem finds a tree connecting the set S with cost at most twice the minimum cost of any Steiner Tree.*

Today we want to show that we can use this algorithm as a subroutine in an multicast admission control problem. As defined above we will use $cost(S)$ to denote the minimum cost of a Steiner Tree containing node set S, and let $cost_{\mathcal{A}}(S)$ denote the cost of the Steiner Tree found by the algorithm in the theorem.

The algorithm is almost the same as before: a request set S is accepted if the cost of the tree found $cost_{\mathcal{A}}(S)$ less than 2^k, and if accepted the set is connected along the tree found by the algorithm.

We will show the same sequence of lemmas as before. Here we only discuss the proof if it is different from the corresponding proof from last time.

Lemma 2 *No edge is used by more than k paths.*

Let $\mathcal{A}$ denote the set of requests accepted by the algorithm, and $\mathcal{O}$ denote the set of requests accepted by the optimum solution. The proof of the first two lemmas is same as before. We use $c(e)$ to denote the final cost of edge e, $cost_c(S)$ to denote the minimum cost of the Steiner Tree containing nodes S subject to costs c.

Lemma 3 $2^k|\mathcal{A}| \geq \sum_e c(e) - m.$

Lemma 4 $\sum_{S\in\mathcal{O}} cost_c(S) \leq k \sum_e c(e)$

The only lemma that is affected by the fact that the algorithm might not find the minimum cost Steiner tree is the last one. We have the following version with a factor of 2 weaker bound.

Lemma 5 $2^k|\mathcal{O} \setminus \mathcal{A}| \leq 2k \sum_e c(e).$

Proof. Let $cost(S)$ denote the cost of the Steiner Tree found by the algorithm at the time S was considered. This cost is at most twice the minimum cost of a Steiner tree at the time. As before edge costs are monotone increasing, and so $cost_c(S)$ is more than the minimum cost, $cost(S)$ at the time S is considered. Putting this together we get that $cost_{\mathcal{A}}(S) \leq 2cost(S) \leq 2cost_c(S)$.

The rest of the proof follows as before: All sets S in $\mathcal{O} \setminus \mathcal{A}$ have been rejected by the algorithm, hence have $cost_{\mathcal{A}}(S) \geq 2^k$. So we have that $cost_c(S) \leq 2 \cdot 2^k$.

We use the above lemma and this observation for the proof.

$$2^k|\mathcal{O} \setminus \mathcal{A}| \leq \sum_{S\in\mathcal{O}\setminus\mathcal{A}} cost_{\mathcal{A}}(S) \leq 2k \sum_e c(e).$$

■

Using these lemmas we can prove the main theorem as before.

Theorem 3 $|\mathcal{O}| \leq (4k+2)|\mathcal{A}|.$

Admission control with capacities. We will not discuss this version in detail. For simplicity we go back to point to point connections here (though the same idea also works for multicast). Here there are two changes: requests for connecting pairs (s,t) come with a given bandwidth demand $dem(s,t)$, and edges have a capacity $cap(e)$. Instead of requirements on the value of k we need an assumption that k the ratio of a capacity to a demand is at least $\log_2 m$:

$$k = \frac{min_e cap(e)}{max_{(s,t)} dem(s,t)} \geq log_2 m.$$

The cost function has to change somewhat. Let $f(e)$ denote the total demand routed through edge e: $f(e) = \sum dem(s,t)$ where the sum is taken over accepted (s,t) pairs whose path is using edge e. Note that when all demands are 1 this is the number of paths using edge e. The cost of edge e will be

$$c(e) = m^{\frac{f(e)}{cap(e)}} = 2^{f(e)\frac{\log_2 m}{cap(e)}}.$$

Again note that when all capacity is the same value $k = \log_2 m$ than this cost function is the same as what we used before. We will not include a detailed analysis of this algorithm here. The main outline is the same as before, except the proofs need a bit more calculus due to the more involved definition of the costs.