College of Computer Science
Northeastern University
Wireless Networks CS G250

Lecture 14
October 29 2008
Lecturer: R. Sundaram

Capacity of Wired Networks

1 Max-flow Min-cut Theorem

Let's take a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Let s be the source and t the sink. What is the maximum flow that can be pushed from s to t?

The Ford-Fulkerson algorithm computes the maximum flow in a flow network. It was published in the 60s. The Ford-Fulkerson algorithm works only if all weights are integers. Otherwise it is possible that the algorithm won't converge to the maximum value.

The maximum flow in a network is equal to the capacity of the minimum cut in the network and it was proved in the Max-flow Min-cut Theorem.

We define a cut as being a partition of the vertices of G such that s is on one side of the cut and t is on the other side of the cut. The capacity of the cut is the sum of the edges crossing the cut. This concept is depicted in Figure 1.

Before we proceed, just a little bit of terminology:

- augmenting path: a path from source s to sink t, along which we can send more flow
- residual graph: a graph that consists of edges that can admit more flow

The basic idea of the Max-flow Min-cut Theorem is the following:

- find the augmenting path from s to t
- send the maximum possible flow M along this path
- "remove" M
- repeat the above 3 steps

We see that "remove" is in between quotes. For that let's take a look at Figure 3. The maximum flow we can send through this graph is equal

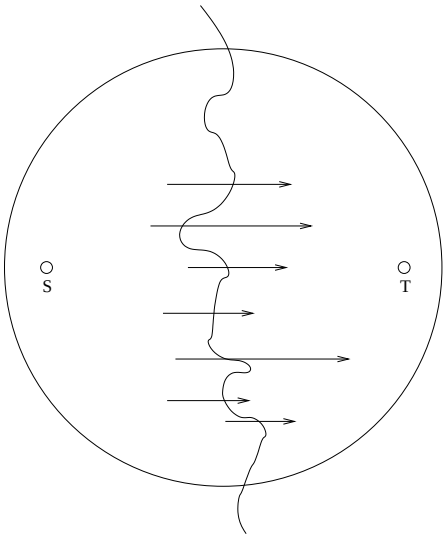

Figure 1: Max-flow Min-cut

to 200. Suppose we choose the path from s to t going through the edge of capacity 10 as the first augmenting path. Through this augmenting path we can send a maximum flow of 10. If we literally remove the edge, then what we're left with is the graph on the right of Figure 3. The maximum flow we can send through the remaining graph is 180. Therefore by removing the edge of capacity 10, we can only send a maximum flow of 190, instead of 200. Thus, literally removing the edge is not a good idea.

What we can do instead is the following. For every augmenting path we can find the maximum flow M we can send. Then for every edge $E(u, v)$ that's part of the augmenting path, remove M from the edge $E(u, v)$ and add an edge of opposite orientation $E(v, u)$ and of capacity M to the graph. This will give us the residual graph. This is depicted in Figure 4. In this new approach we can send successfully a maximum flow of 200.

The augumenting path algorithm is guaranteed to stop because each augmentation increases the flow by at least 1 and the flow is upper-bounded by the mincut. When all capacities are integers, then we have a finite number of steps.

Let's look at what happens at terminantion. If t is still reachable from s, that means we can still augument it. At terminantion, for mincut all edges should look like in Figure 2. There should be no edges going from left to right in the mincut. If there is an edge going left to right through the

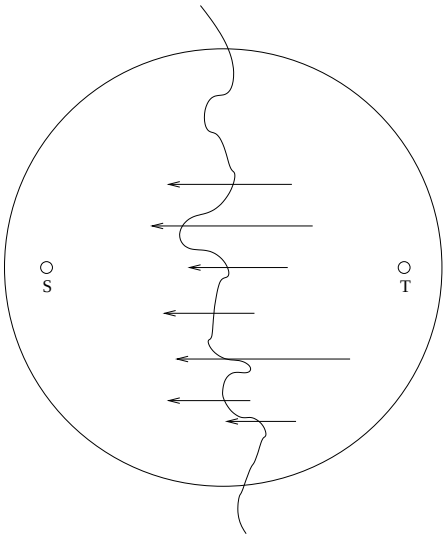

Figure 2: Max-flow Min-cut at termination

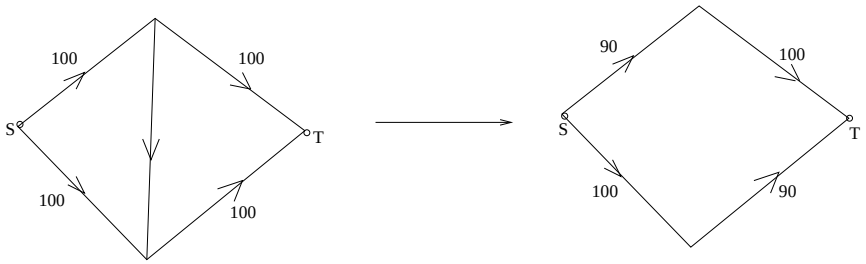

Figure 3: Augumenting path - incorrect strategy

mincut, then what that implies is that there is a path going from S to the boundary (cut) and there is a path from the boundary to t, meaning there is a still path from s to t.

A cut like the one in Figure 1 for example went down to 0. What that means is that the maximum flow is greater than or equal to the capacity of any cut.

We knew from before that maximum flow is less than or equal to the capacity of any cut. In conclusion, maximum flow is equal to mincut.

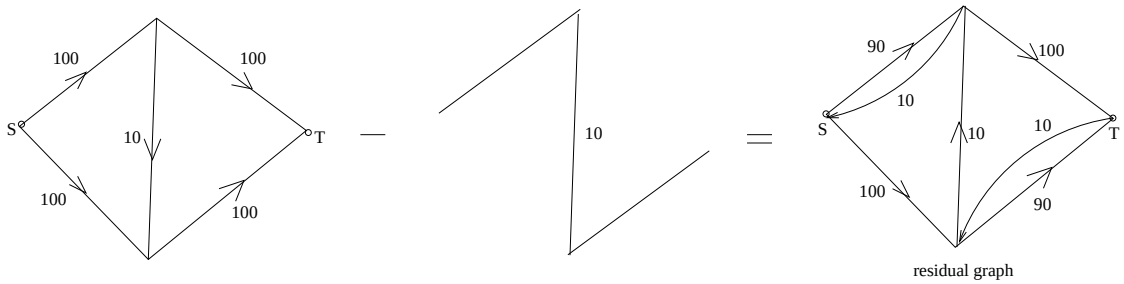

Figure 4: Augumenting path - correct strategy

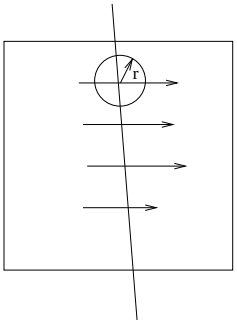

Figure 5: Peraki-Servetto Argument

2 Proof of Gupta-Kumar

The Gupta-Kumar Theorem says that if we are given a square cell of edge 1 and we place n people in this cell, then the bandwidth per conversation as $n \to \infty$ goes to 0. Mathematically, this can be expressed as: $\lim_{n\to\infty} \frac{poly\log(n)}{\sqrt{n}} = 0$.

The Peraki-Servetto argument is depicted in Figure 5. The total link capacity is $O\left(n^2r^3\right)$. The radius r is $\sqrt{\frac{\log n}{n}}$. There are roughly $\frac{n}{4}$ conversations going through the cut. The bandwidth per conversation is $\frac{n^2r^3}{n} = nr^3 = \frac{\log n^{\frac{3}{2}}}{\sqrt{n}}$.

Now let's look at the orginal Gupta-Kumar theorem depicted in Figure 6. There are n pairs: $s_1 - t_1$, $s_2 - t_2$, ..., $s_n - t_n$, where $s_k - t_k$ means that s_k wants to talk to t_k.

The distance between $s_i - t_i \geq L$. By distance we mean the number of hops.

Thus, the bandwidth per conversation is less than $\frac{nd}{nL} = \frac{d}{L}$, where nd is

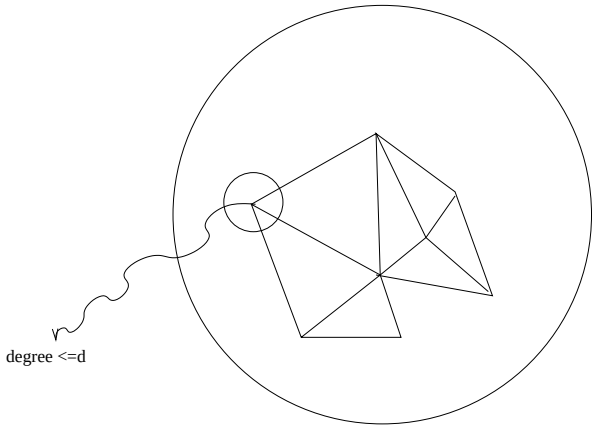

Figure 6: Gupta-Kumar Argument

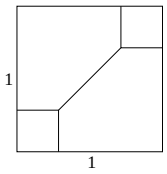

Figure 7: Maximum distance between 2 points

the total volume.

So, $\sum_{conversations} fL = nfL$. nfL is the total volume, n the number of conversations, f is the flow we can send and L is the minimum distance between any $s_i - t_i$.

Therefore, $nfL \leq \frac{nd}{2}$.

Let's take a look at Figure 7. What's the maximum distance between 2 points in a square cell of edge 1? The answer is obviously $\sqrt{2}$.

On average we have that $E[L] \geq \frac{1}{4} \Rightarrow E[L] \geq \Omega(1)$.

$E[L] \geq \Omega(\frac{1}{r})$

In an area of $\pi \frac{\log n}{n}$ we have $\pi \log n$ points. The degree of a node is $\log n$.

Thus, $\frac{d}{L} = dr = \log n \sqrt{\frac{\log n}{n}} = \frac{\log n^{\frac{3}{2}}}{\sqrt{n}}$.