Dr. Christian Scheideler
Johns Hopkins University

Baltimore, Sept. 24 2003

# Theory of Network Communication

Fall 2003

## Assignment 3

**Problem 6** (4 points)**:**
Consider the algorithm given in Figure 1. Show that for any flow problem with demands $d_i = 1$ for every $i$ that has a feasible flow solution with paths of length at most $L$ when using demands $d_i' = (1+\epsilon)$, it holds that the algorithm with sufficiently large queues never has to delete any flow.
**Hints:** The original Awerbuch-Leighton (AL) algorithm maximizes $\sum_i f_i(\Delta_i(e) - f_i)$, which represents exactly the amount by which the potential at the queues at $e$ drops when moving a normalized flow of $f_i$ for each commodity $i$. When using instead the rule in Figure 1, we have to distinguish between two cases:

- $\max_i \Delta_i(e) > 2$: Then argue that $f_i$ in Figure 1 can be set to 1. So the potential drop is $(\Delta_i(e) - 1) \geq (\sum_j f_j(\Delta_j(e) - f_j)) - 1$, where the $f_j$ are chosen as in the original AL-Algorithm.
- $\max_i \Delta_i(e) \leq 2$: Then simply use the fact that $\max_i \Delta_i(e)/d_i \geq \sum_j f_j(\Delta_j(e) - f_j)$ where the $f_j$ are chosen as in the original AL-Algorithm.

Use the arguments above to show that in any case, the discrete Awerbuch-Leighton Algorithm achieves a potential drop that is at most an additive 2 worse at any edge than the potential drop achieved by the original Awerbuch-Leighton Algorithm. Use this insight to adapt the proof in the lecture notes for the original Awerbuch-Leighton Algorithm to show that one can bound the queue sizes. This should mostly be just cut-and-paste. **Note that the proof has been updated in the lecture notes!!**

**Problem 7** (6 points)**:**
Implement the Discrete Awerbuch-Leighton algorithm in the Spheres simulation environment for the graph in Figure 2. Output the maximum observed queue size (obtained after at least 200 rounds).

**Discrete Awerbuch-Leighton Algorithm**:

At each node $u$:

1. Distribute newly injected flow evenly among the buffers $Q_i(e)$, i.e. distribute it so that afterwards for every $i$, $\bar{q}_i(e)$ is the same for every edge $e$ leaving $u$.
2. For every edge $(u, v)$, select any $i$ with maximum $\Delta_i(u, v)$. If this is negative, no flow is sent. Otherwise, compute $f_i = \min\{1,\ \Delta_i(u, v)/2\}$ and send a flow of $c(e) \cdot f_i$ from $Q_i(u, v)$ to $Q_i(v, u)$.
3. Receive the transmitted flow and absorb flow that reached its destination.
4. Rebalance the queue heights so that for every $i$, $\bar{q}_i(e)$ is the same for every edge $e$ leaving $u$.

Figure 1: The Discrete Awerbuch-Leighton algorithm.

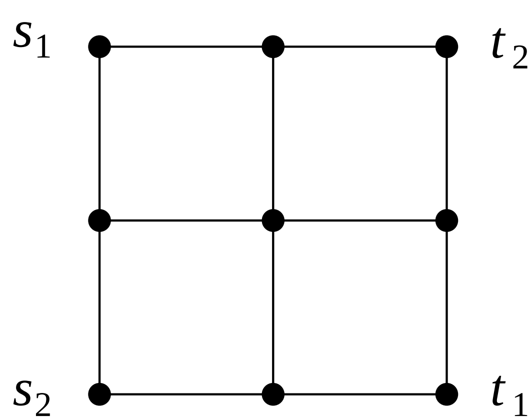


Figure 2: An example graph for the Discrete Awerbuch-Leighton algorithm. Both source nodes inject a flow of 1 into the graph at every time step. All edges have capacity 1.