# 4 Network Flows III – Multicommodity Flows

In this section we study multicommodity flows. Consider an undirected graph $G = (V, E)$ with an assignment of non-negative capacities to the edges, $c : E \to \mathbb{R}^+$. A *multicommodity flow* instance on $G$ is a set of ordered pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. Each pair $(s_i, t_i)$ represents a *commodity* with source $s_i$ and target $t_i$. The objective is to maximize the amount of flow traveling from the sources to the corresponding destinations, subject to the capacity constraints. The problem comes in two flavors. In the first, called the *maximum multicommodity flow* problem, the total flow, summed over all commodities, is to be maximized. The second is called the *maximum concurrent flow* problem. Here, for each commodity $(s_i, t_i)$ a non-negative demand $d_i$ is specified. The objective is to maximize the *fraction* of the demand that can be shipped simultaneously for all commodities. Both the maximum multicommodity flow problem and the maximum concurrent flow problem can be solved in polynomial time using linear programming.

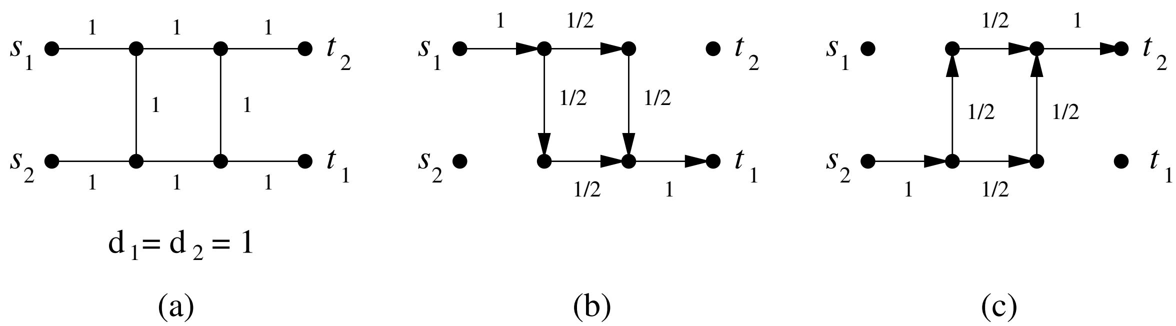


Figure 1: Solution to a 2-commodity flow problem (a). The routing of the first commodity is shown in (b) and the second commodity is shown in (c).

## 4.1 The flow number

Next we introduce an important parameter called flow number that will allow us to study the efficiency of multicommodity flow algorithms. First, some definitions.

Consider a maximum concurrent flow problem. For every multicommodity flow $f$ consisting of flow $f_i$ for each commodity $i$, the *concurrent flow value* of $f$ is defined as $\min_i |f_i|/d_i$. The *concurrent max-flow* of a multicommodity flow instance $I$ is defined as the maximum over all concurrent flow values of feasible multicommodity flow solutions for $I$. There are two important classes of maximum concurrent flow problems:

In a *product multicommodity flow problem* (PMFP) we associate a non-negative *weight* $\pi(u)$ with each node $u \in V$. The demand for the commodity between nodes $u$ and $v$ is then set to be $\pi(u) \cdot \pi(v)$. A *balanced multicommodity flow problem* (BMFP) is a multicommodity flow problem in which the sum of the demands of the commodities originating and the commodities terminating in a node $v$ is equal to $c(v)$ for every $v \in V$.

Suppose we have a network $G = (V, E)$ with arbitrary non-negative edge capacities. Given a concurrent multicommodity flow problem with feasible solution $\mathcal{S}$, let the *dilation* $D(\mathcal{S})$ of $\mathcal{S}$ be defined as the length of the longest flow path in $\mathcal{S}$ and the *congestion* $C(\mathcal{S})$ of $\mathcal{S}$ be defined as the inverse of its concurrent flow value (i.e., the congestion says how many times the edge capacities would have to be increased in order to satisfy the demands of all commodities when using the same set of paths). Let $\mathcal{B}$ be the PMFP in which $\pi(v) = c(v)/\sqrt{c(V)}$ for every node $v$, that is, each pair of nodes $(v, w)$ has a commodity of demand $c(v) \cdot c(w)/c(V)$. Note that $\mathcal{B}$ is also a BMFP.

**Definition 4.1 ([4])** *The* flow number $F(G)$ *of a network $G$ is defined as the minimum over all feasible solutions $\mathcal{S}$ of $\mathcal{B}$ of* $\max\{C(\mathcal{S}), D(\mathcal{S})\}$.

In the case that there is no risk of confusion, we will simply write $F$ instead of $F(G)$. Note that the flow number of a network is invariant to a scaling of the capacities.

The flow number can be computed exactly via linear programming in polynomial time. Another advantage of the flow number is that, as shown by the next theorem, it can be applied to much more general multicommodity flow problems than just the one that defines it.

**Theorem 4.2** *For any network $G$ with flow number $F$ and any instance $I$ of the BMFP for $G$, there is a feasible solution for $I$ with congestion and dilation at most $2F$.*

**Proof.** The idea is to decompose $I$ into two multicommodity flow problems: for every commodity $i$ with source $s_i$ and destination $t_i$, the first problem $I_1$ has commodities $i_u$ from $s_i$ to $u$ for all $u \in V$ with demands $d_{i_u} = d_i \cdot c(u)/c(V)$, and the second problem $I_2$ has commodities $i'_u$ from $u$ to $t_i$ for all $u \in V$ with demands $d_{i'_u} = d_i \cdot c(u)/c(V)$. It follows from these definitions that for every commodity $i$ from the original problem, the total demand of corresponding commodities leaving $s_i$ in $I_1$ is $d_i$ and the total demand of corresponding commodities destined for $t_i$ in $I_2$ is $d_i$ as well. Moreover, for every node $u \in V$ the fraction of the demand of commodity $i$ directed to $u$ in $I_1$ is equal to the fraction of commodity $i$ leaving $u$ in $I_2$. Hence, gluing feasible flow solutions for $I_1$ and $I_2$ together and reducing their flow values by a factor of 2 would give a feasible flow for the original problem.

Combining commodities with the same source-destination pair into one, both of the flow problems $I_1$ and $I_2$ can be turned into PMFPs with $\pi(v) = c(v)/\sqrt{c(V)}$ for every node $v$, because for any pair $v, w \in V$, the total demand of the commodities with source $v$ and destination $w$ in $I_1$ is equal to

$$\sum_{i:\, s_i = v} \frac{d_i \cdot c(w)}{c(V)} = \frac{c(v) \cdot c(w)}{c(V)} = \pi(v) \cdot \pi(w) \ ,$$

and in $I_2$ is equal to

$$\sum_{i:\, t_i = w} \frac{d_i \cdot c(v)}{c(V)} = \frac{c(v) \cdot c(w)}{c(V)} = \pi(v) \cdot \pi(w) \ .$$

Therefore, both $I_1$ and $I_2$ represent the multicommodity flow problem $\mathcal{B}$ underlying the definition of the flow number. Thus, both $I_1$ and $I_2$ have a feasible solution with congestion and dilation at most $F$. Hence, the original problem $I$ has a feasible solution with congestion and dilation at most $2F$, which proves the claim. $\square$

With techniques similar to those used in the proof of Theorem 5.0.3 in [5] one can also prove the following result.

**Theorem 4.3** *On average over all BMFPs $I$, the minimum* $\max\{C(\mathcal{S}), D(\mathcal{S})\}$ *over all feasible solutions $\mathcal{S}$ of $I$ is* $\Omega(F)$.

Hence, the flow number truthfully captures the problem of routing BMFPs in networks. Using Theorem 4.2, we prove another powerful result, called *Shortening Lemma*, that shows that the flow number allows to convert arbitrary multicommodity flow solutions into solutions with short flow paths.

**Theorem 4.4 (Shortening Lemma [4])** *Suppose we are given a network with flow number $F$. Then, for any $\epsilon \in (0, 1]$ and any feasible multicommodity flow $f$, there exists a feasible multicommodity flow $f'$ with flow values $|f'_i|$ of at least $|f_i|/(1+\epsilon)$ for every commodity $i$ that uses paths of length at most $2 \cdot F(1 + 1/\epsilon)$.*

**Proof.** Given a flow solution $\mathcal{S}$, let $\mathcal{S}' \subseteq \mathcal{S}$ consist of all paths from $\mathcal{S}$ that are longer than $L$, for $L = 2 \cdot F/\epsilon$. We are going to shorten the paths in $\mathcal{S}'$ at the cost of slightly decreasing the satisfied demand of each commodity.

For a path $p \in \mathcal{S}'$ between $s_p$ and $t_p$, let $a_{p,1} = s_p, a_{p,2}, \cdots, a_{p,L}$ denote its first $L$ nodes and $b_{p,1}, \cdots, b_{p,L-1}, b_{p,L} = t_p$ its last $L$ nodes and let $f_p$ be the flow value along $p$. Then the set $\mathcal{U} = \bigcup_{p \in \mathcal{S}'} \bigcup_{i=1}^{L} \{a_{p,i}, b_{p,i}, f_p\}$ is (a subset of) an instance of the BMFP. By Theorem 4.2, there exists a feasible solution $\mathcal{P}$ to $\mathcal{U}$ with flow value at least $1/(2F)$ consisting of paths of length at most $2F$. We are going to combine the initial and final parts of the long paths in $\mathcal{S}'$ with these "shortcuts" in $\mathcal{P}$ to obtain the desired short solution.

First, decrease the flows along all paths $p \in \mathcal{S}$ by a factor of $1/(1+\epsilon)$ so that we have room to accommodate new, short paths for the paths in $\mathcal{S}'$. These short paths are constructed in the following way:

For every path $p \in \mathcal{S}'$, we replace $p$ by $L$ flow systems $S_{p,i}$, $i = 1, \cdots, L$. Each flow system $S_{p,i}$ consists of two parts:

1. the flow paths between $a_{p,i}$ and $b_{p,i}$ in $\mathcal{P}$ corresponding to the request $\{a_{p,i}, b_{p,i}, f_p\}$ from $\mathcal{U}$, now with a flow of $f_p/(L(1+\epsilon))$, and
2. $f_p/(L(1+\epsilon))$ units of flow between $a_{p,1}$ and $a_{p,i}$ along $p$, and $f_p/(L(1+\epsilon))$ units of flow between $b_{p,i}$ and $b_{p,L}$ along $p$.

For each $i$, the length of each path in the subsystem $S_{p,i}$ is at most $L + 2 \cdot F$, and $f_P/(L(1+\epsilon))$ units of flow are shipped along each path system $S_{p,i}$. Summed over all $i = 1 \ldots L$, we have $f_p/(1+\epsilon)$ units of flow between $s_p = a_{p,1}$ and $t_p = b_{p,L}$, which is as high as the original flow through $p$ reduced by $1/(1+\epsilon)$. Hence, we can replace $p$ by the systems $S_{p,i}$ without changing the amount of flow from $s_p$ to $t_p$.

Now, it holds for every edge $e$ that the flow traversing $e$ due to the paths in $\mathcal{S}$ is at most $c(e)/(1+\epsilon)$, and due to the shortcuts in $\mathcal{P}$ is at most

$$\sum_{p \in \mathcal{P}:\, e \in p} \frac{f_p}{L(1+\epsilon)} \leq \frac{2F}{L(1+\epsilon)} \cdot c(e) = \frac{\epsilon \cdot c(e)}{1+\epsilon} \ ,$$

since

$$\sum_{p \in \mathcal{P}:\, e \in p} \frac{f_p}{2F} \leq c(e) \ .$$

Thus, the flows in $\mathcal{S}$ and $\mathcal{P}$ sum up to at most $c(e)$ for an edge $e$. Therefore, the modification yields a feasible solution satisfying the desired properties. $\square$

## 4.2 An algorithm for feasible multicommodity flow problems

We start with a concurrent multicommodity flow problem $\mathcal{P}$ in which the commodities $i$ have demands $d_i$ so that there is a feasible solution for $\mathcal{P}$ even when increasing all demands to $(1+\epsilon)d_i$ for some $\epsilon > 0$. Before we present a distributed algorithm for this problem, we need some notation.

In the following, when we speak about edges, we always mean directed edges (i.e. we specify in which direction to use an undirected edge). In every node, every outgoing edge has a queue for every commodity. Given an edge $(u,v)$ and a commodity $i$, let $Q_i(u,v)$ denote the queue at node $u$ buffering flow of commodity $i$ for edge $(u,v)$ and let $q_i(u,v)$ denote the amount of flow in $Q_i(u,v)$. Furthermore, let $\bar{q}_i(u,v) = q_i(u,v)/(d_i \cdot c(u,v))$ and let $\Delta_i(u,v) = \bar{q}_i(u,v) - \bar{q}_i(v,u)$.

The concurrent multicommodity flow problem will be solved by interpreting it as a flow problem in which source $s_i$ pumps a demand of $d_i$ into the system in each step. Any flow algorithm that is able to deliver the injected flow with bounded queues without deleting any flow must come up with flow strategies for the injected flows so that one can construct a feasible flow for the original problem. Ideally, the algorithm would even converge to such a flow, but for multicommodity flows this has not been proven yet. The algorithm we will consider in this section is given in Figure 2 (see also [1]).

**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 $(u,v)$, find flows $f_i \geq 0$ so that
$$\sum_i f_i \cdot (\Delta_i(u,v) - f_i)$$
is maximized under the constraint that
$$\sum_i d_i \cdot f_i \leq 1 \ .$$
For each commodity $i$, send a flow of $c(e) \cdot d_i \cdot f_i$ 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 2: The Awerbuch-Leighton algorithm.

Although the maximization problem in the Awerbuch-Leighton algorithm looks complicated, there is actually an easy solution to it. It can be shown that for an optimal solution, there exists an $s \geq 0$ such that
$$f_i = \max\left\{\frac{\Delta_i(e) - s \cdot d_i}{2}\ ,\ 0\right\}$$

for all $i$. The optimal value of $s$ is the minimum $s \geq 0$ so that

$$\sum_i d_i \max\left\{\frac{\Delta_i(e) - s \cdot d_i}{2}\ ,\ 0\right\} \leq 1\ .$$

This value can be found via binary search.

The following theorem shows that if the queues are sufficiently large, then the Awerbuch-Leighton algorithm never has to delete any injected flow, as desired. Recall that $m$ denotes the number of edges in the network.

**Theorem 4.5** *For any flow problem with $K$ commodities that, when augmenting the demands by a $(1+\epsilon)$ factor, still has a feasible solution that uses paths of length at most $L$, the Awerbuch-Leighton algorithm with queues that can accommodate a flow of $\bar{q}_i(e) = 2(1+\epsilon)\sqrt{m}KL/\epsilon$ never has to delete any flow.*

**Proof.** Let $f$ be any feasible flow for the flow problem with demands $(1+\epsilon)d_i$. We will compare the performance of the Awerbuch-Leighton algorithm (or short AL-algorithm) against this flow. For this we use a potential method. For any edge $e$ and commodity $i$, we define

$$\Phi_i(e) = \frac{c(e)}{2} \cdot \bar{q}_i^2(e)\ .$$

The potential of the entire system, $\Phi$, is simply

$$\Phi = \sum_{e,i} \Phi_i(e)\ .$$

To simplify the proof for bounding this potential, we assume (without loss of generality) that $c_{\min}$, the minimum edge capacity, is equal to 1 (this can be achieved by appropriate scaling) and that all edge capacities are integral multiples of $c_{\min}$ (this can be achieved by simple rounding, which has a negligible effect if $c_{\min}$ is sufficiently small). Then we can view every edge $(u,v)$ as representing $c(u,v)$ edges of capacity 1 from $u$ to $v$. Evenly distributing the flow in $Q_i(u,v)$ among the queues for these new edges simplifies the definition of $\bar{q}_i(e)$ to $\bar{q}_i(e) = q_i(e)/d_i$ and $\Phi_i(e)$ to $\Phi_i(e) = \bar{q}_i^2(e)/2$ for every new edge $e$ and commodity $i$ without changing the original potential.

For this multi-graph $G'$ (i.e. a graph with multiple edges between the same pair of nodes), consider any decomposition of $f$ into a system $\mathcal{P}$ of flow paths $p$ of length at most $L$ (and flow values $f(p)$ so that $\sum_{p \in \mathcal{P}} f(p) = |f|$). Let $p$ be any one of these flow paths, and let $i$ be the commodity associated with $p$. Consider any edge $e \in p$. The AL-algorithm selects flows to be forwarded along $e$ so as to maximize the potential drop. This can be seen by representing the potential $\Phi_i(e)$ as an integral:

$$\Phi_i(e) = \int_{h=0}^{\bar{q}_i(e)} h\, dh\ .$$

Hence, the potential $\Phi_i(e)$ can be interpreted as the sum of all heights of all flow pieces (normalized by $d_i$) stored in $Q_i(e)$. When sending a (normalized) flow of $f_j$ of commodity $j$ from $u$ to $v$, every (normalized) piece of commodity $j$ achieves a drop in height of exactly $\Delta_j(u,v) - f_j$, i.e. we get a potential drop for commodity $j$ of $f_j(\Delta_j(u,v) - f_j)$. Choosing the $f_1, \ldots, f_k$ that maximize

$\sum_j f_j(\Delta_j(u,v) - f_j)$ (subject to the constraint that the flow sent does not exceed the edge capacity) therefore maximizes the potential drop.

Thus, the potential drop achieved by the AL-algorithm at edge $e$ is at least as high as the potential drop achieved by sending flow according to $\mathcal{P}$ at $e$. Hence, as a worst case for the potential drop of the AL-algorithm, we consider the case that the AL-algorithm actually sends flow as prescribed by $\mathcal{P}$. In this case, given that $\mathcal{P}$ sends a flow of $\varphi$ for commodity $i$ across $e$, the potential drop caused by commodity $i$ at $e = (u,v)$ is equal to

$$\begin{aligned} & (\bar{q}_i^2(u,v)/2 - (\bar{q}_i(u,v) - \bar{\varphi})^2/2) + (\bar{q}_i^2(v,u)/2 - (\bar{q}_i(v,u) + \bar{\varphi})^2/2) \\ = \ & \bar{\varphi}(\bar{q}_i(u,v) - \bar{q}_i(v,u)) - \bar{\varphi}^2 \end{aligned}$$

where $\bar{\varphi} = \varphi/d_i$. This potential drop can be split among the flow paths $p'$ across $e$ belonging to $i$ by simply replacing $\bar{\varphi}$ by $\bar{f}(p')$, where $\bar{f}(p') = f(p')/d_i$. Hence, we can assign to the flow path $p$ in $\mathcal{P}$ a potential drop of

$$\bar{f}(p)(\bar{q}_i(u,v) - \bar{q}_i(v,u)) - \bar{f}(p)\bar{\varphi} \geq \bar{f}(p)(\bar{q}_i(u,v) - \bar{q}_i(v,u)) - (1+\epsilon)\bar{f}(p) \ .$$

Summing up over all edges of $p$, $p$ creates a potential drop of at least

$$\sum_{(u,v)\in p} (\bar{f}(p)(\bar{q}_i(u,v) - \bar{q}_i(v,u)) - (1+\epsilon)\bar{f}(p)) \geq \bar{f}(p)\bar{q}_i - (1+\epsilon)L \cdot \bar{f}(p)$$

where $\bar{q}_i$ is the normalized amount of commodity $i$ in the queues of the source of commodity $i$. Notice that for every commodity $i$, the sum of the $\bar{f}(p)$ over all of its flow paths $p$ is equal to $(1+\epsilon)d_i/d_i = (1+\epsilon)$. Hence, the total potential drop due to the movement of flow in steps 2 and 3 of the AL-algorithm is at least

$$\left(\sum_i (1+\epsilon)\bar{q}_i\right) - (1+\epsilon)L \cdot (1+\epsilon)K \ .$$

On the other hand, the potential increase caused by injecting new flow at step 1 of the AL-algorithm is at most

$$\sum_i \bar{q}_i \ .$$

Step 4 of the AL-algorithm can only decrease the potential. Hence, the overall potential increase in one round of the AL-algorithm is at most

$$-\epsilon \sum_i \bar{q}_i + (1+\epsilon)^2 L \cdot K \ . \tag{1}$$

This value is guaranteed to be negative (i.e. the potential decreases) if

$$\sum_i \bar{q}_i > (1+\epsilon)^2 L \cdot K/\epsilon \ .$$

Since flow is only sent downwards, it must hold that $\bar{q}_i$ is the maximum queue size for commodity $i$ in any queue of the system. Because there are $2m$ queues of each commodity in the system and according to (1), $\Phi$ increases by at most $(1+\epsilon)^2 L \cdot K$ in any step, the potential is limited to

$$\Phi \leq 2m \cdot ((1+\epsilon)^2 L \cdot K/\epsilon)^2/2 + (1+\epsilon)^2 L \cdot K$$

where $m$ is the number of edges in the system. In the worst case, all of this potential may be concentrated in a single queue. Hence, the maximum value a $\bar{q}_i(e)$ can attain is bounded by $2\sqrt{m} \cdot (1+\epsilon)^2 L \cdot K/\epsilon$. $\square$

Using a more complicated variant of the Awerbuch-Leighton algorithm with normalized queue sizes $\bar{q}_i(e) = q_i(e)/d_i$ [2], one can even reduce the bound on the queue sizes to $\bar{q}_i(e) = O((1+\epsilon)L \cdot (\log(K^2 m/\epsilon^2))/\epsilon)$. With the help of the Shortening Lemma, Theorem 4.5 immediately implies the following result:

**Corollary 4.6** *For any flow problem with $K$ commodities in a network with flow number $F$ that, when augmenting the demands by a $(1+\epsilon)$ factor, still has a feasible solution, the Awerbuch-Leighton algorithm with queues that can accommodate a flow of $\bar{q}_i(e) = \Theta((1+\epsilon)^2\sqrt{m}KF/\epsilon)$ never has to delete any flow.*

In the next week, we will see that for some networks the flow number can be very low (e.g., $O(\log n)$) and therefore the queue size required for the routing is acceptable.

## 4.3 The maximum multicommodity flow problem

For the maximum multicommodity flow problem, we just have to slightly modify the Awerbuch-Leighton algorithm. Since we do not have explicit demands, we set $d_i = C = \max_v c(v)$ for every commodity $i$, i.e. $\bar{q}_i(e) = q_i(e)/(C \cdot c(e))$. Every source node $s_i$ of commodity $i$ injects $C$ units of flow for commodity $i$ into the system at each time step, which is certainly an upper bound on what the system can deliver. Furthermore, each queue now has a limited size, i.e. it can only store a limited amount of flow. Any flow exceeding its limit is simply deleted. Using this strategy, we arrive at the algorithm shown in Figure 3.

This algorithm has the following performance:

**Theorem 4.7** *For any maximum multicommodity flow problem with $K$ commodities, the bounded Awerbuch-Leighton algorithm with queues that can accommodate a flow of up to $\bar{q}_i(e) = 2L/\epsilon$, where $f_i$ if the flow value of commodity $i$ in an optimal solution, delivers a flow that is at most a $(1+\epsilon)$-factor away from an optimal maximum flow.*

**Proof.** We only sketch the proof here. We will use a potential function $\Phi$ where the individual $\Phi_i(e)$'s are defined similar to the proof above, but this time we distinguish between two types of flows: flow that is still "on track", and flow that "got behind" an optimal flow strategy. Only flow that got behind is considered in the potential. Concentrating only on flow that is part of an optimal concurrent flow solution, one can show the following properties:

- The deletion of a flow of $f$ decreases $\Phi$ by at least $f \cdot L/\epsilon$.
- Routing a flow of $f$ along a path of length at most $L$ increases $\Phi$ by at most $f \cdot L$.

Hence, after $t$ rounds, $\Phi \leq t \cdot \lambda \cdot L - \delta \cdot L/\epsilon$, where $\lambda$ is the max-flow of the flow problem and $\delta$ is the total amount of relevant flow deleted by the bounded AL-algorithm. Since $\Phi$ must be positive at any time, it follows that $\delta \leq \epsilon\lambda \cdot t$, which proves the theorem. $\square$

**Bounded 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 $u$. *Delete any excess flow that cannot be stored in the buffers.*

2. For every edge $(u, v)$, find flows $f_i \geq 0$ so that

$$\sum_i f_i \cdot (\Delta_i(u, v) - f_i)$$

is maximized under the constraint that $\sum_i f_i \leq 1/C$. For each commodity $i$, send a flow of $c(e) \cdot C \cdot f_i$ 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 3: The bounded Awerbuch-Leighton algorithm.

We note that it is not clear whether the algorithm actually converges to an approximate maximum flow solution. But on average, it will deliver approximately the same amount of flow as an optimal solution can deliver, which allows to compute an approximate maximum flow out of the combination of flows used by the algorithm over time.

### 4.4 The maximum concurrent flow problem

One may also adapt the AL-algorithm to solve any maximum concurrent flow problem. Simply start with the original demands. Every time we observe for some edge $e$ and commodity $i$ that $\bar{q}_i(e) > 2\sqrt{m}(1+\epsilon)K \cdot L/\epsilon$, we reduce the flow everywhere by a factor of $(1+\epsilon)$ (by broadcasting a notification to all nodes via a minimum spanning tree, for example). We assume that this results in a $(1 + \Theta(\epsilon))$-approximation of the concurrent max-flow.

### 4.5 Min-cost multicommodity flows

Finally, we note that also distributed algorithms for min-cost multicommodity flow problems have been considered. See [3] and the references therein for further information.

## References

[1] B. Awerbuch and F. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proc. of the 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 459–468, 1993.

[2] B. Awerbuch and F. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proc. of the 26th ACM Symp. on Theory of Computing (STOC)*, pages 487–496, 1994.

[3] A. Kamath, O. Palmon, and S. Plotkin. Fast approximation algorithm for minimum cost multicommodity flow. In *Proc. of the 7th ACM/SIAM Symp. on Discrete Algorithms (SODA)*, 1995.

[4] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *ACM/SIAM Symp. on Discrete Algorithms (SODA)*, 2002.

[5] C. Scheideler. *Universal Routing Strategies for Interconnection Networks*, volume 1390 of *Lecture Notes in Computer Science*. Springer, 1998.