J Comb Optim (2011) 22:426–437
DOI 10.1007/s10878-010-9321-y

Why locally-fair maximal flows in client-server networks perform well

Kenneth A. Berman · Chad Yoshikawa

Published online: 22 May 2010

Abstract Maximal flows reach at least a $1/2$ approximation of the maximum flow in client-server networks. By adding 1 additional time round to any distributed maximal flow algorithm we show how this 1/2-approximation can be improved on bounded-degree networks. We call these modified maximal flows 'locally fair' since there is a measure of fairness prescribed to each client and server in the network. Let $N = (U, V, E, b)$ represent a client-server network with clients U, servers V, network links E, and node capacities b, where we assume that each capacity is at least one unit. Let $d(u)$ denote the b-weighted degree of any node $u \in U \cup V$, $\Delta = \max\{d(u)|u \in U\}$ and $\delta = \min\{d(v)|v \in V\}$. We show that a locally-fair maximal flow f achieves an approximation to the maximum flow of $\min\{1, \frac{\Delta^2-\delta}{2\Delta^2-\delta\Delta-\Delta}\}$, and this result is sharp for any given integers δ and Δ. This results are of practical importance since local-fairness loosely models the steady-state behavior of TCP/IP and these types of degree-bounds often occur naturally (or are easy to enforce) in real client-server systems.

Keywords Distributed flow algorithms · Oblivious routing · Network algorithms · Maximal flow

1 Introduction

Using a maximal flow to approximate the maximum flow is attractive for client-server networks since (i) it can be computed without global information, (ii) it can be com-

K.A. Berman · C. Yoshikawa (✉)
Computer Science Department, University of Cincinnati, Cincinnati, OH 45221, USA
e-mail: chad.yoshikawa@uc.edu

K.A. Berman
e-mail: ken.berman@uc.edu

puted quickly, and (iii) it reaches a 1/2-approximation of the maximum throughput. For example, maximal flows are the basis for the 'Restricted Adversary' routing (Awerbuch et al. 2005) and the 'Aggressive Increase' routing algorithm (Annexstein et al. 2007)—both of which are locally-adaptive algorithms that reach constant-competitiveness in $O(\log(\Delta))$ distributed time rounds where Δ is the maximum client degree in the network. Other examples of distributed maximal flow algorithms include the 'BipartiteMatch' and 'LowDegreeMatch' distributed algorithms (Hanckowiak et al. 1998) which compute a maximal flow on n-node client-server networks in $polylog(n)$ and $O(\Delta)$ time rounds, respectively.

In general, a maximal flow on an arbitrary client-server network cannot do better than a 1/2-approximation of the maximum flow. However, maximal flows generally outperform this approximation in practice. For example, it was shown (Annexstein et al. 2007) that the aforementioned 'Restricted Adversary' and 'Aggressive Increase' distributed maximal-flow algorithms reach better than 75% and 90%, respectively, of the optimal throughput on a variety of bipartite graphs (Cherkassky et al. 1998). On the other hand, there exist extremal graphs for which these algorithms can only attain the 1/2 lower bound of maximal flows. The question is, under what set of conditions does a maximal flow perform well?

In this paper, we show that we can do much better than a 1/2-approximation if the network satisfies certain degree conditions and the maximal flow f is *locally-fair*. More precisely, let $N = (G = (U \cup V, E), b)$ be a client-server network, where G is a bipartite graph with node set $U \cup V$ and edge set E representing the topology of the network, i.e., $uv \in E$ if and only if a communication link has been established between client u and server v, and b is a capacity weighting of the nodes $U \cup V$. (For a client u, $b(u)$ represents the demand of client u.) A flow f is a weighting of E with nonnegative real numbers such that, for each $x \in U \cup V$, $f(x) \le b(x)$, where $f(x) = \sum_{xy\in E} f(xy)$. The *size* of a flow f, which we denote by *size*(f), is the total flow over all the edges, i.e., *size*$(f) = \sum_{e\in E} f(e)$. A maximum flow is one that maximizes *size*(f). A maximal flow is one such that for each $uv \in E$, either client u or server v is saturated, where a node $x \in U \cup V$ is *saturated* with respect to f, if $f(x) = b(x)$.

1.1 Locally-fair flows

First consider the case where each client sends at most one unit of flow, i.e., $b(u) = 1, u \in U$. In this case, if a server were to reserve an equal portion of its capacity for each neighborhood client u, then u would receive its "fair" share of the server's capacity equal to $b(v)/d(v)$, where $d(v)$ is the degree of v. We consider flows f that are "client locally fair" in the sense that, either client u is saturated (is fully satisfied), or u sends a flow to each neighborhood server v, which is at least as great as its fair share of v's capacity (i.e., at least $b(v)/d(v)$). This definition extends to the general case, where clients may have non-unit capacities, except that fair share becomes $b(u)b(v)/d(v)$, where $d(v)$ becomes the b-weighted degree $d_b(v)$. For $x \in U \cup V$, the *b-weighted degree of* x is the total capacity over all nodes in the neighborhood of x, i.e., $d_b(x) = \sum_{xy\in E} b(y)$. For convenience, in the remainder of this paper we will denote $d_b(x)$ simply by $d(x)$.

In this paper we consider flows, which we call "locally-fair", that are not only locally fair to the clients as described above, but satisfy the additional condition that, if a client u is saturated and does not send its fair share of v's capacity to server v, then v receives at least its fair share of u's demand (i.e., at least $b(u)b(v)/d(u)$).

Definition 1 A flow f is *locally-fair* if, for each edge $uv \in E$, $f(uv) \geq \min\{\frac{b(u)b(v)}{d(u)}, \frac{b(u)b(v)}{d(v)}\}$ and if the client u is unsaturated, $f(uv) \geq \frac{b(u)b(v)}{d(v)}$.

The following proposition shows that locally-fair flows always exist.

Proposition 1 *Let $N = (G = (U \cup V, E), b)$ be a client-server network. Then, there always exists a locally-fair flow f.*

Proof Let μ be the edge-weighting given by $\mu(uv) = \min(\frac{b(u)b(v)}{d(u)}, \frac{b(u)b(v)}{d(v)})$. Clearly, μ satisfies the capacity constraints on the nodes, i.e., is a flow in the network. For any client that is not saturated, i.e. a client u for which $\sum_{uv \in E} \mu(uv) < b(u)$, increase the weight on each of its outgoing edges uv up to $b(u)b(v)/d(v)$ (or less) until the client is saturated or until all edges have been exhausted. The resulting flow f is locally-fair. □

Based on the proof of Proposition 1 a locally-fair flow can be computed in an distributed setting using only a few communication steps.

Locally-fair flow (LFF) algorithm

Input: Distributed client-server network $N = (G = (U \cup V, E), b)$
Output: Distributed locally-fair flow f in N

```
1. for each server $v \in V$ do in parallel
       server $v$ sends the values $b(v)$ and $d(v)$ to each client $u$ in
       its neighborhood
2. for each client $u \in U$ do in parallel
       client $u$ assigns initial flow values $\min\{b(u)b(v)/d(u), b(u)b(v)/d(v)\}$
       for each edge $uv$
       while client $u$ not saturated
           client increases flow values on each edge $uv$ up to the
           value $b(u)b(v)/d(v)$
```

Note that any extension of f is necessarily locally-fair. Using more rounds, the initial flow generated by algorithm *LFF* can be extended to a locally-fair *maximal* flow by computing a maximal flow g using any distributed maximal flow algorithm (Annexstein et al. 2007; Hanckowiak et al. 1998) with respect to the residual capacities (where the residual capacity of node $x \in U \cup V$ is $b(x) - f(x)$) and increasing the flow on edge e by $g(e)$. On an n-node client-server network, these distributed maximal flow algorithms are $O(\Delta)$ and $O(polylog(n))$, respectively. We can also obtain a locally-fair maximal flow by repeatedly calling *LFF* with residual capacities.

Locally-fair maximal flow (LFMF) algorithm

Input: Distributed client-server network $N = (G = (U \cup V, E), b)$
Output: Distributed locally-fair maximal flow f in N

```
1. call LFF to obtain a distributed flow $f$
2. while there exist an edge $uv$ where both $u$ and $v$ are unsaturated
       call LFF with capacity $b(x)$ replaced with residual capacity
       $r(x) = b(x) - f(x)$,
           $x \in U \cup V$, to obtain a distributed flow $f_r$ and add this flow to $f$
```

After the first round each client u sends a flow of $f(uv)$ to each neighborhood server v and increases this flow by $f_r(uv)$ after each subsequent round. Algorithm *LFMF* is actually a special case of an aggressive increase algorithm as described in Annexstein et al. (2007), under the slightly more general condition that clients may have non-unit capacities, and with the additional demand that $f(uv) \geq \mu(uv)$ for all $uv \in E$. Algorithm *LFMF* performs at most Δ_V rounds where Δ_V denotes the maximum (un-weighted) degree of a server. To see this observe that the flow f output by *LFF* has the property that each server s is either saturated, or at least one neighborhood client of s is saturated. This follows from the fact that unsaturated clients send their fair-share to s and therefore s would be saturated by f if all neighborhood clients were unsaturated.

The following proposition shows that Algorithm *LFMF* is very close to maximal after $O(\log \Delta_U)$ rounds, where Δ_U denotes the maximum (un-weighted) degree of a client.

Proposition 2 *After performing* $2\log_2(\Delta_U/\epsilon) + 4$ *rounds the flow* f *generated by algorithm* LFMF *is a* $\frac{1}{1+\epsilon}$*-approximation to a locally-fair maximal flow, i.e.,* $size(f) = (\frac{1}{1+\epsilon})size(g)$ *for some locally-fair maximal flow* g.

The proof is given in Appendix.

In this paper we consider the following natural degree constraints, which are practical in a distributed setting and enforceable without global communication, for which algorithm *LFMF* achieves much better than a 1/2-approximation.

(Δ, δ) Dual-Bounded Network A network such that every client has b-weighted degree at most Δ and every server has b-weighted degree at least δ.

Our main result is the following theorem.

Theorem 1 *Let* $N = (G = (U \cup V, E), b)$ *be a* (Δ, δ)*-dual-bounded network, where* $b(x) \geq 1$ *for all* $x \in U \cup V$, *and let* f *be any locally-fair maximal flow. If* $\delta \geq \Delta$ *then* f *is a maximum flow; otherwise,* f *achieves an approximation to a maximum flow of at least* $\frac{\Delta^2-\delta}{2\Delta^2-\delta\Delta-\Delta}$. *Further, this result is sharp for any given integers* δ *and* Δ.

Given a fraction p, it immediately follows from Theorem 1 that a p-approximation is achieved if

$$\delta \geq \frac{\Delta^2(2p-1)-\Delta p}{\Delta p-1}$$

For example, suppose $\Delta = 10$. Then a .9-approximation is achieved if $\delta \geq 8.875$.

2 Related work on tight bounds for maximal flows

As far as we know, Theorem 1 is the first tight bound on maximal flows besides the aforementioned 1/2-bound. There is work (Cardinal et al. 2005) which is similar in spirit in that a bound is given for the approximation that a maximal flow yields for the vertex covering problem.

Similar to our work is that of Czygrinow et al. (2004) which presented an algorithm for extending a maximal matching to achieve a 2/3-approximation to the maximum matching. In this work, the authors used the method of removing short augmenting paths in order to improve the matching size. However, instead of requiring 1 additional round of distributed computation the algorithm in that work required an additional $polylog(n)$ additional time rounds where n is the number of nodes in the graph.

3 Dual bounds

In this section, we prove Theorem 1 which relates the size of a locally-fair flow to the maximum flow in terms of the dual-bounds Δ and δ.

First, we provide the following lemma which guarantees that every server $v \in N$ receives at least $b(v)\min(1, \delta/\Delta)$ flow.

Lemma 1 *In a locally-fair flow every server $v \in V$ receives flow at least $\min\{\frac{\delta}{\Delta}b(v), b(v)\}$.*

Proof Pick any server v. If $d(v) \geq \Delta$ then each incoming edge uv carries flow of $b(u)b(v)/d(v)$ and the total incoming flow to v is $b(v)$ and the server is saturated. Otherwise, each incoming edge uv carries flow at least $b(u)b(v)/\Delta$ which sum to at least $b(v)\min(1, \delta/\Delta)$. □

Corollary 1 *In a network N, if $\delta \geq \Delta$, then a locally-fair flow is maximum and has size equal to the total capacity of the servers, s.*

The corollary follows immediately from Lemma 1.

Now we will prove Theorem 1 which is repeated here:

Claim 1 *Let $N = (G = (U \cup V, E), b)$ be a (Δ, δ)-dual-bounded network, where $b(x) \geq 1$ for all $x \in U \cup V$, and let f be any locally-fair maximal flow. If $\delta \geq \Delta$ then f is a maximum flow; otherwise, f achieves an approximation to the maximum flow of at least $\frac{\Delta^2-\delta}{2\Delta^2-\delta\Delta-\Delta}$. Further, this result is sharp for any given integers δ and Δ.*

Proof Let f be any locally-fair maximal flow having size $\phi = \sum_{e\in E} f(e)$. If $\delta \geq \Delta$, then by Corollary 1, f is a maximum flow. Therefore, we can assume that $\delta < \Delta$ in the rest of the proof.

We will prove the theorem by lower-bounding the flow of first the servers and then the clients. First, the flow of the servers is examined. For convenience, we partition the clients U and servers V into the following sets:

– W the unsaturated clients
– X the saturated clients (also the client-complement of W)
– Y the neighborhood of W, i.e. $N(W)$
– Z the server-complement of Y

For any set S, we denote as f_S the size of the flow entering (or leaving) the set S, i.e., $f_S = \sum_{x\in S, xy\in E} f(xy)$. For convenience, hereafter we refer to flow size as "flow" whenever the usage is unambiguous. Using this notation, the flow entering the servers is given by:

$$\phi = f_V = f_Y + f_Z$$

Note that all servers in $Y = N(W)$ are necessarily saturated since (1) clients in W are unsaturated by definition and (2) any maximal flow induces a vertex cover of saturated clients and saturated servers. Thus, the flow size can be rewritten as:

$$f_V = b(Y) + f_Z$$

where $b(Y)$ is the total capacity of the servers in set Y, i.e. $b(Y) = \sum_{y\in Y} b(y)$.

Using Lemma 1, every server $z \in Z$ must receive flow at least $\frac{\delta}{\Delta} b(z)$. Summing over all $z \in Z$, then, the flow size f_Z is at least $b(Z)\frac{\delta}{\Delta}$. Thus, the flow size into the servers, f_V, satisfies the following inequality:

$$f_V \geq b(Y) + b(Z)\frac{\delta}{\Delta} \tag{1}$$

Now, we examine the flow leaving the client set U. The flow leaving the clients, f_U, is given by:

$$\phi = f_U = f_X + f_W = b(X) + f_W$$

since all nodes in X are saturated by definition.

The flow f_W is equivalent to the flow entering the server set Y restricted to the clients in W. By definition of a locally-fair flow, since every client in W is unsaturated then each client $w \in W$ will always send flow at least $\frac{b(w)b(y)}{d(y)}$ for each $y \in N(w)$.

First, let us define the notation that $d_W(y)$ is the b-weighted degree of node y restricted to the set W, e.g. $d_W(y) = \sum_{w\in N(y)\cap W} b(w)$. Using this notation we have the following inequality:

$$f_W \geq \sum_{y\in Y} b(y)\frac{d_W(y)}{d(y)} = \sum_{y\in Y} b(y)\frac{d_W(y)}{d_W(y) + d_X(y)} \geq \sum_{y\in Y} b(y)\frac{1}{1 + d_X(y)}$$

where the last inequality holds since all b-values are at least 1 and Y is the neighborhood of W.

Thus, we can rewrite the bound on total client flow, f_U, as:

$$f_U \geq b(X) + \sum_{y \in Y} b(y) \frac{1}{1 + d_X(y)} \geq b(X) + \frac{b(Y)^2}{b(Y) + \sum_{y \in Y} b(y) d_X(y)}$$

where the second line follows from the arithmetic mean-harmonic mean (AM-HM) inequality.

The sum $\sum_{y \in Y} b(y) d_X(y)$ can be bounded using the definitions of Δ and δ:

$$\sum_{x \in X} b(x)\Delta \geq \sum_{x \in X} (b(x) d_Y(x) + b(x) d_Z(x)) = \sum_{y \in Y} b(y) d_X(y) + \sum_{z \in Z} b(z) d_X(z)$$

This result implies that $\sum_{y \in Y} b(y) d_X(y) \leq b(X)\Delta - b(Z)\delta$ and so provides an upper bound on the sum.

So, in a locally-fair flow, the client flow is bounded by the following inequality:

$$f_U \geq b(X) + \frac{b(Y)^2}{b(Y) + b(X)\Delta - b(Z)\delta} \tag{2}$$

Combining this result with the previous bound for the server flow, we have these two inequalities:

$$\phi = f_U \geq b(X) + \frac{b(Y)^2}{b(Y) + b(X)\Delta - b(Z)\delta}$$
$$\phi = f_V \geq b(Y) + b(Z) \frac{\delta}{\Delta}$$

Setting $w = b(W)$, $x = b(X)$, $y = b(Y)$, and $z = b(Z)$, by Lemma 2 (see Appendix) the value of the locally-fair maximal flow f is at least $\min(1, \frac{\Delta^2 - \delta}{2\Delta^2 - \delta\Delta - \Delta})$ times the maximum flow m and the first statement of the theorem is proved.

For the second statement of the theorem, that the given bound is tight, consider the following. For any combination of integers Δ, δ create a unit capacity network and divide the clients and servers each into two sets W, X and Y, Z. For convenience, denote $w = |W|$, $x = |X|$, $y = |Y|$, and $z = |Z|$.

Set $w = y$ and $x = z$. Let each client $u \in W$ have one outgoing edge, each client $u \in X$ have Δ outgoing edges, each server $v \in Y$ have Δ in-edges, and each server $v \in Z$ have δ in-edges. Match each client $u \in W$ to a unique server $v \in Y$ and let each client $u \in X$ have $\Delta - \delta$ edges matched to the servers in Y and the remaining δ edges matched to servers in W such that there is a perfect matching in the graph and the non-matching edges are evenly distributed across the servers in the applicable server sets.

Given any value for y, we set $x = y\frac{\Delta - 1}{\Delta - \delta}$, and the other two parameters (w and z) are determined by the equality constraints above. (The values x and y can be scaled so that they are integers.) The size of y is determined by the fact that $x + y = m$ which implies that $y = m\frac{\Delta - \delta}{2\Delta - \delta - 1}$. See Fig. 1 which shows an example.

Define a flow f such that each client $u \in W$ sends $1/\Delta$ flow along its single edge, and each client $u \in X$ sends $1/\Delta$ flow along each of its Δ edges. Clearly, f is

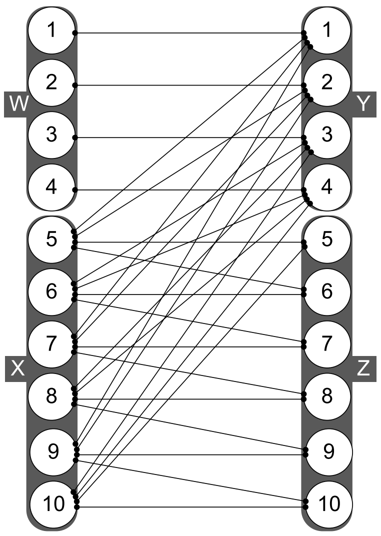

Fig. 1 Worst-case graph with $y = 4$, $\Delta = 4$, and $\delta = 2$. The maximum flow is 10 while a locally-fair maximal flow only provides $\frac{\Delta^2-\delta}{2\Delta^2-\delta\Delta-\Delta}10 = 7$

a locally-fair flow and every server $v \in Y$ is saturated while every server $v \in Z$ is unsaturated and receives δ/Δ flow.

The maximum flow in the graph is equal to $x + y = m$, since G contains a perfect matching. Thus, the ratio of the locally-fair flow f to the maximum flow is $(y + z(\delta/\Delta))/(y + x) = (\Delta^2 - \delta)/(2\Delta^2 - \Delta - \delta\Delta)$ since $z = x$. This is equal to our lower bound and thus the second statement of the theorem is proved. □

4 Practicality

In a network of clients and servers it is well-known that TCP/IP, or any additive-increase multiplicative-decrease (AIMD) algorithm, converges to a flow whereby each unsaturated client receives bandwidth from each server at least equal to its fair share (Kurose and Ross 2000). The definition of 'Locally-fair' in this paper is a slightly stronger requirement since it has the additional constraint that every edge contain at least a minimal amount of flow. However, in our preliminary experiments, TCP/IP typically has higher aggregate bandwidth than the equivalent 'locally-fair' assignment would predict on the same network. So, it may be that the bounds in this paper are lower-bounds for max-min fair algorithms such as TCP/IP. This remains future research.

The degree-bounds in this paper are either locally enforceable or may occur naturally in practice. Networks with low Δ and high δ values can be achieved in several ways. It is common for clients to have restricted access to servers (or peers) in order to limit the advantage of any single client over any other (Qiu and Srikant 2004). This is a side-effect of the max-min fairness provided by TCP/IP to clients—the more streams the more bandwidth a client can steal from others. Furthermore, maintaining a set of streams becomes impractical as the number of streams gets too large: TCP/IP will reset the sender's TCP window on any stream which is idle for more than a time-out period (Stevens 1997). This effectively limits the number of outgoing streams a single modern computer can maintain. (There is no equivalent idle-detection mechanism on the receiver side.) So it is clear that limiting the maximum client degree, Δ, should be practical and simple to implement in software distributed to clients.

Maintaining the minimum server degree, δ, can be managed in one of two ways. First, in a randomly-constructed network where every client has exactly Δ connections, the minimum server degree is within a constant factor of Δ with high probability. With a balls-and-bins analysis (omitted here) it can be shown that as long as $\Delta \geq 4c^2 \ln(2|V|)$ then $\delta \geq (1 - 1/c)\Delta$ w.h.p. Thus, in a randomly-constructed network, the two parameters Δ and δ can be coupled as long as $\Delta \in \Omega(\ln(|V|)$. Another way to increase the value of δ is via a systems-level solution of server gossiping—clients can be exchanged from low-degree servers to high-degree servers by periodic server-to-server contact.

5 Open issues and further research

Other types of maximal flows are also of interest: "Locally-greedy maximal flows" (LGMF) and "Locally-proportional maximal flows" (LPMF). In LGMF saturating clients satisfy server demands from largest to smallest value while in LPMF these saturating clients satisfy server demands in relative proportion.

For the unit-capacity case where the maximum client degree is 2, it can be shown that LGMF always reaches at least $5/6$ of the maximum flow and LPMF always reaches at least $3/4$ of the maximum flow. (LFMF has a tight bound of $3/4$ of the maximum flow for this case.) Furthermore, these bounds are tight (proofs are omitted). For general graphs, it is an open problem to determine how well these flows perform.

Appendix

Lemma 2 *Given constants m, δ, and Δ, the solution to the non-linear program below is a value at least* $\min(1, \frac{\Delta^2-\delta}{2\Delta^2-\delta\Delta-\Delta})m$.

$$\begin{array}{ll} \min & f(x,y,z) \\ \text{s.t.} & f(x,y,z) \geq x + \frac{y^2}{y+x\Delta-z\delta} \\ & f(x,y,z) \geq y + z\frac{\delta}{\Delta} \\ & m \geq x, \ \ y \geq 0 \\ & x + y \geq m \\ & y + z \geq m \\ & \Delta > \delta \geq 1 \end{array}$$

Proof We have:

$$f(x,y,z) \geq x + \frac{y^2}{y + x\Delta - z\delta} \geq x + \frac{y^2}{y(\delta+1) + x\Delta - m\delta}$$

and

$$f(x,y,z) \geq y + z\frac{\delta}{\Delta} \geq y\left(1 - \frac{\delta}{\Delta}\right) + m\frac{\delta}{\Delta}$$

since $z \geq m - y$ and both functions are increasing in terms of z.

Now we have two functions of x and y:

$$g_1(x,y) = x + \frac{y^2}{y(\delta+1) + x\Delta - m\delta} \tag{3}$$

$$g_2(x,y) = y\left(1 - \frac{\delta}{\Delta}\right) + m\frac{\delta}{\Delta} \tag{4}$$

where $f(x,y,z) \geq \min(g_1(x,y), g_2(x,y))$.

Instead of searching the space of all possible pairs x, y which satisfy the constraint $x + y \geq m$, we can instead search the space along all lines of constant $x + y$. In our case, since $2m \geq x + y \geq m$, we will search along lines where $x + y = m + c$ for an arbitrary constant c such that $m \geq c \geq 0$. If we prove that the lemma holds for an arbitrarily chosen value of c, then we have proven the lemma.

First, choose any value of c therefore $y = m - x + c$. Substituting this value for y, we can now create two functions in terms of x only:

$$h_1^c(x) = x + \frac{(m - x + c)^2}{(m - x + c)(\delta+1) + x\Delta - m\delta} \tag{5}$$

$$h_2^c(x) = (m - x + c)\frac{(\Delta - \delta)}{\Delta} + \frac{\delta}{\Delta}m \tag{6}$$

with

$$f(x,y,z) \geq \min(g_1(x,y), g_2(x,y)) = \min(h_1^c(x), h_2^c(x)).$$

Setting $h_1^c(x) = h_2^c(x)$ and solving for x we obtain two possible solutions: $x_1 = \frac{c\delta}{\delta - \Delta}$ and $x_2 = \frac{m(\Delta-1)+c(\Delta-\delta-1)}{2\Delta-\delta-1}$. The first intersection point, x_1, yields a value of m (when $c = 0$) or is an invalid point of intersection (when $c > 0$) since x is non-negative.

Now we consider x_2. Clearly, $h_2^c(x,y)$ is a decreasing function of x. We will now show that $h_1^c(x,y)$ is monotonically increasing after the intersection point x_2 and therefore the minimum flow value is always at least the flow value $h_1^c(x_2) = h_2^c(x_2)$ which occurs at the intersection point x_2.

First, note that $h_1^c(x)$ is positive and defined in the interval $x \in [c, m]$ since $x + y \geq m$ and $\Delta > \delta \geq 1$.

Now, we show that the first derivative of $h_1^c(x)$ at x_2 is non-negative.

$$\frac{d}{dx}h_1^c(x_2) = \frac{\Delta^2 - 3\Delta + \delta + 1}{\Delta^2}$$

For all possible values of δ and Δ:

$$\frac{\Delta^2 - 3\Delta + \delta + 1}{\Delta^2} \geq 0$$
$$\iff \Delta^2 - 3\Delta + \delta + 1 \geq 0$$
$$\iff \Delta^2 - 3\Delta + 2 \geq 0$$

where the last line results from the fact that $\delta \geq 1$. The last line is true since $\Delta^2 - 3\Delta + 2$ is an increasing function of Δ and $\Delta \geq 2$.

Now, we show that the second derivative of $h_1^c(x)$ is positive. The second derivative of $h_1^c(x)$ is $2\frac{(m\delta-(c+m)\Delta)^2}{(c(1+\delta)+m+(\Delta-\delta-1)x)^3}$ Examining this function, we see that this second derivative is always positive for $x, c \geq 0$. Thus, for the domain $x > 0$, there are no points of inflection and any extrema of $h_1^c(x)$ can only be minimums by the second derivative test. Thus, since (1) $h_1^c(x)$ is either a local minimum or increasing at the intersection point x_2, (2) there are no maximums, and (3) $h_1^c(x)$ is well-defined on all points $x \geq 0$ then $h_1^c(x)$ must be monotonically increasing after the point of intersection x_2. Thus, the flow value at the point of intersection of $h_1^c(x)$ and $h_2^c(x)$ must be a global minimum of the composite function $\max(h_1^c(x), h_2^c(x))$.

Now, plugging x_2 back into h_1^c we obtain a flow value of

$$\frac{(\Delta^2 - \delta)m + c\Delta(\Delta - \delta)}{\Delta(2\Delta - \delta - 1)}$$

Since $c \geq 0$ and $\Delta > \delta$, then this flow value is at least

$$\frac{(\Delta^2 - \delta)m}{2\Delta^2 - \delta\Delta - \Delta}$$

Since we used an arbitrary value of c, the equations hold for all values of c and thus the lemma is proved. □

Now we will prove Proposition 2 which is repeated here:

Proposition 2 *After performing* $2\log_2(\Delta_U/\epsilon) + 4$ *rounds the flow* f *generated by algorithm* LFMF *is a* $\frac{1}{1+\epsilon}$*-approximation to a locally-fair maximal flow, i.e., size*$(f) = (\frac{1}{1+\epsilon})$*size*$(g)$ *for some locally-fair maximal flow* g.

Proof We employ the following lemma.

Lemma 3 *Let* f *be the flow produced by algorithm* LFMF *after performing* $2j$ *rounds. Then, for each server* v*, either* $r(v) \leq \frac{b(v)}{2^j})b(v)$ *or* $d_r(v) \leq \frac{d_b(v)}{2^j}$*, where* r *denotes the residual capacity.*

Proof Consider any server v and let $N(v)$ denote the clients in the neighborhood of v. Let S and T denote the subsets of $N(v)$ that are saturated and unsaturated, respectively, after a (single) execution of the algorithm *LFF*. Then, $b(S) + b(T) = d_b(v)$. If

$b(S) \geq b(T)$ then clearly $d_r(v) \leq d_b(v)/2$. On the other hand, if $b(S) \leq b(T)$, then since, each node $u \in T$ is unsaturated, $f(uv) \geq b(u)b(v)/d_b(v)$. But, this means that the flow $f(v)$ into v is at least $\sum_{u \in T} f(uv) \geq b(u)b(v)/d_b(v) = b(T)b(v)/d_b(v) \geq (d_b(v)/2)b(v)/d_b(v) = b(v)/2$, so that $r(v) \leq b(v)/2$. It follows that after every two applications of *LFF*, either the residual capacity of the neighborhood of v (i.e., $d_r(v)$) has been cut by at least one-half or the residual capacity of v has been cut by at least one-half. Thus, after $2j$ rounds, either the (original) capacity of v or the (original) capacity of the neighborhood of v has been reduced by a factor of $\frac{1}{2^j}$. The lemma follows. □

Now, let Q denote the set of all servers v, such that $r(v) \leq \frac{b(v)}{2^j}$, let R denote the complement set of servers, and let P denote the set of clients that are adjacent to a server in R. By Lemma 3, $d_r(v) \leq \frac{d_b(v)}{2^j}$, for all $v \in R$. Since, each client has degree at most Δ_U, we have:

$$r(P) \leq \sum_{v \in R} d_r(v) \leq \frac{1}{2^j} \sum_{v \in R} d_b(v) \leq \frac{\Delta}{2^j} b(P)$$

It follows that $r(P) \leq \frac{\Delta}{2^{j-1}} f(P)$, where $f(P)$ denotes the total flow out of P. Further, it follows from the definition of Q that $r(Q) \leq \frac{1}{2^{j-1}} f(Q) \leq \frac{\Delta}{2^{j-1}} f(Q)$, where $f(Q)$ denotes the total flow into Q. Since $P \cup Q$ is a vertex cover, i.e., every edge is incident to one node in $P \cup Q$, we can increase f to a maximal flow g in a way that only increases the flow out of nodes in P or the flow into nodes in Q. But, this means that the size of the additional flow added to f to obtain g is at most $r(P) + r(Q) \leq \frac{\Delta}{2^{j-1}}(f(P) + f(Q) \leq \frac{2\Delta}{2^{j-1}} size(f)$. Hence, $size(g) \leq (1 + \frac{\Delta_U}{2^{j-2}}) size(f)$. Proposition 2 follows by setting $j = 2 + \log_2 \Delta_U/\epsilon$. □

References

Annexstein F, Berman KA, Strunjas S, Yoshikawa C (2007) Maximizing throughput in minimum rounds in an application-level relay service. In: Workshop on algorithm engineering & experiments (ALENEX)

Awerbuch B, Hajiaghayi MT, Kleinberg RD, Leighton T (2005) Online client-server load balancing without global information. In: SODA '05: proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, pp 197–206

Cardinal J, Labbé M, Langerman S, Levy E, Mélot H (2005) A tight analysis of the maximal matching heuristic. In: Computing and combinatorics: 11th annual international conference, COCOON 2005. Lecture notes in computer science, vol 3595. Springer, Kunming, pp 701–709

Cherkassky BV, Goldberg AV, Martin P, Setubal JC, Stolfi J (1998) Augment or push: a computational study of bipartite matching and unit-capacity flow algorithms. J Exp Algorithmics 3:8. http://doi.acm.org/10.1145/297096.297140

Czygrinow A, Hańćkowiak M, Szymańska E (2004) Distributed algorithm for approximating the maximum matching. Discrete Appl Math 143(1–3):62–71. doi:10.1016/j.dam.2003.10.004

Hanckowiak, Karonski, Panconesi (1998) On the distributed complexity of computing maximal matchings. In: SODA: ACM-SIAM symposium on discrete algorithms, a conference on theoretical and experimental analysis of discrete algorithms. citeseer.ist.psu.edu/article/hanckowiak97distributed.html

Kurose JF, Ross KW (2000) Computer networking: a top-down approach featuring the Internet package. Addison-Wesley, Boston

Qiu D, Srikant R (2004) Modeling and performance analysis of bittorrent-like peer-to-peer networks. URL citeseer.ist.psu.edu/qiu04modeling.html

Stevens W (1997) TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, Internet Engineering Task Force