IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Selection of Paths in HDR

Dror Rawitz, Adrian Segall

CCIT Report #919
February 2018

THE ANDREW & ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING
TECHNION—ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 3200003, ISRAEL

CCIT Report #919 February 2018

Selection of Paths in HDR

Dror Rawitz* Adrian Segall†

February 5, 2018

Abstract

In this report we consider the problem of Path Selection in HDR, a Hysteresis Driven Routing Protocol for EnHANTS - networks with Energy Harvesting units. HDR requires selection of two non-necessarily disjoint paths for each source. The paths are alternately used by the source to transmit data. Path selection in HDR leads to the problem of 2-splittable flow in graphs, that has been previously treated in the literature. In this report we model our problem as a 2-splittable flow problem, show that in general it is NP-Hard and provide algorithms for Fair Path Selection under the assumption of a uniform network.

1 Introduction and Model

Consider a wireless network, where one or more *sources* transmit data to a single collection point, referred to as the *sink*. The network consists of energy harvesting nodes, referred to as *relaying nodes*. Since we are interested in transmission limited by the harvested energy, the wireless links are assumed to be have unlimited data transmission capacities.

The Harvesting Rates of the relaying nodes are possibly time varying and unknown. They limit the packet transmission rate from the sources to the sink. As we are interested in selecting routes that allow maximum transmission over the network, the sources and the sink should not limit data transmission. Thus, we assume energy-unlimited sources and sink. In addition, the sink is assumed to be land based and to possess a powerful transmitter, used in order to transmit control messages that are heard directly by all sources and by all relaying nodes. Moreover, the sink is assumed to have a processor that is able to run the routing algorithm and to send commands for its implementation. The considered network is referred to in [MGS+15] as EnHANTs (Energy Harvesting Networked Tags).

Given the limited amount of energy at the relaying nodes, classical routing algorithms, like AODV [PR99] and DSR [JMB01] and their later versions, are not applicable. Much simpler algorithms need to be developed. In HDR [Seg15], [ELSY17], [LS] it has been proposed to employ for each source two fixed *Routing Paths* to the sink and to split traffic between the two. HDR employs alternately the two paths, that are preferably, but not necessarily, disjoint. Nodes that

*School of Engineering, Bar Ilan University, Ramat Gan, Israel, dror.rawitz@biu.ac.il. Supported in part by the Israel Science Foundation (grant no. 497/14).

†Viterbi Dept. of Electrical Engineering, Technion, Haifa, Israel, segall@ee.technion.ac.il.

are common to the two Routing Paths are referred to as *Common-Nodes*. The non-Common node with minimum energy on each Routing Path is referred to as the *Critical Node*.

The nodes on the *inactive path* that are not Common-Nodes harvest energy, thus their energy level goes up. The nodes on the *active path*, including the Common-Nodes, both harvest and spend energy, the latter on data packet transmissions. In normal operation, the spending rate is higher than the harvesting rate, thus their energy level goes down. The algorithm switches from one Routing Path to the other when the energy in the critical node of the inactive path exceeds by a certain hysteresis threshold the energy in the critical node of the active path. The algorithm also dictates that the sink provides feedback to the sources to adjust their rate according to the current energy situation in the corresponding two paths. In certain cases, the two Routing Paths coincide. HDR is designed to work in this case as well, but obviously there are only input adjustments, with no hysteresis switching.

Relaying a data packet is the main energy consumption action of a relaying node. The other energy consuming actions are transmitting and receiving control packets and overhearing over the wireless channel headers of packets not intended for the node itself. Since control packets and headers of data packets are assumed to be second order, we neglect them for the purpose of selecting Routing Paths, which is the subject of this work. The simulation of the system activity does take those into account. Obviously, packet transmission over each path in the HDR algorithm is time-varying even for fixed input and harvesting rates, depending on which Routing Path is active at any given time. However we consider here only averages over time.

Let $c'(v)$ denote for node v *the average harvesting rate normalized to the energy required to relay a single packet*. System stability requires that the time-average packet relaying rate over each node v is less than or equal to $c'(v)$. Therefore, $c'(v)$ can be viewed as the "capacity" of relaying node v. Thus our problem reduces to the problem of selecting, for each source, two paths, in a network with node capacities $c'(v)$ and infinite link capacities, such that the data flow rate through each node v does not exceed $c'(v)$. The links are assumed to have infinite capacity since we are interested in energy limitations and not limitations due to transmission capacity of the links.

The purpose of this report is to select the Routing Paths. Since Routing Paths must be selected a priori, before operating the network, we need a simple criterion. The criterion employed here is to seek Routing Paths that provide Maximum Total Throughput, with the additional requirement of fairness among sources. Moreover, if there is a choice, we prefer that the two Routing Paths for each source are as disjoint as possible.

This is motivated as follows. For a given network with given fixed source data rates and given paths, the optimal splitting can be found by solving (in MATLAB say) a nonlinear optimization problem whose goal is to maximize the sum of the log of the inputs, given the flow constraints dictated by the node capacities. Since the maximum of the product of several variables whose sum is fixed is achieved when the variables are equal, the solution of this optimization provides fairness among sources. Simulations reported in [ELSY17], as well as later simulations, indicate that the HDR algorithm provides splitting fractions close to the optimal obtained from the above optimization problem. Thus it makes sense to seek Routing Paths and a splitting requirement that provide maximum fair throughput. In addition, simulations show that the more the paths are disjoint, the better performance is obtained (see Section 7 for a simple example).

Since one normally does not know in advance the pattern of the energy harvesting rates, the selection of the Routing Paths will be done assuming a uniform network, namely all relaying nodes

have the same fixed Harvesting Rate. Normalizing to the energy spent by a Relaying Node to relay a data packet, we can view the scenario as a flow graph, where all nodes, except for the source and sink nodes, have the same node capacity. We take this node capacity as our unit of flow, namely $c'(v) = 1$ for all relaying nodes v.

1.1 Related Work

Baier, Köler, and Skutella [BKS05] (see also [Bai03]) and Koch and Spenke [KS06] considered the k-SPLITTABLE FLOW problem which is a variant of NODE 2-SPLITTABLE FLOW (defined below) in which the capacities are on the edges and there is only one source s. In this problem the goal is to find a maximum flow from s to t that has a decomposition into k flow paths. Baier, Köler, and Skutella [BKS05] proved that k-SPLITTABLE FLOW in directed graphs is NP-hard for any constant $k \geq 2$. They also show that it is NP-hard to approximate the 2-SPLITTABLE FLOW problem in directed graphs within a ratio strictly less than 3/2. Later, Koch and Spenke [KS06] extended the NP-hardness result to undirected graphs and also showed that it is NP-hard to approximate the k-SPLITTABLE FLOW problem within a ratio better than 6/5 for any constant $k \geq 2$.

2 Preliminaries

2.1 The Problems

Here we formalize our model and the considered problems. Consider a network $N' = (G', c', S, t)$, where $G' = (V', E')$ is an undirected graph, $c' : V' \to \mathbb{N}$ is a capacity function on the nodes of G', $S = \{s_1, \ldots, s_k\} \subseteq V'$ is a set of k *source* nodes, and $t \in V'$ is the *sink*. We refer to a node $v' \in S \cup \{t\}$ as a *terminal* node and to a node $v' \in V' \setminus (S \cup \{t\})$ as a *relaying* node. We assume that the capacity of relaying nodes is finite and that the capacity of terminal nodes is infinite, namely that $c'(v') < \infty$, for all $v' \in V' \setminus (S \cup \{t\})$ and $c'(v') = \infty$, for all $v' \in S \cup \{t\}$.

We assume that the sources and the sink are not directly connected, i.e., that $(s_i, t) \notin E'$ for every i. The case of *unit capacities* is the case where $c'(v') = 1$, for every relaying node $v' \in V' \setminus (S \cup \{t\})$.

A *flow path* in this network is a pair (P', f'), where P' is a set of nodes along a simple path in G' from some source s_i to the sink and $f' \in \mathbb{R}$. A 2-*splittable flow* $\mathcal{F}'$ consists of two flow paths $(P'_{i,1}, f'_{i,1})$ and $(P'_{i,2}, f'_{i,2})$, for each source s_i, where $P'_{i,1}$ and $P'_{i,2}$ are paths from s_i to t. Note that it may be the case that $P'_{i,1} = P'_{i,2}$. A 2-splittable flow $\mathcal{F}'$ is feasible if $\sum_i \sum_{j:v' \in P'_{i,j}} f'_{i,j} \leq c'(v')$, for every $v' \in V'$. The value $f'(\mathcal{F}')$ of a 2-splittable flow $\mathcal{F}'$ is the total amount of flow that enters t, that is $f'(\mathcal{F}') \triangleq \sum_{i=1}^{k} (f'_{i,1} + f'_{i,2})$. A 2-splittable flow $\mathcal{F}$ is optimal if $f'(\mathcal{F}')$ is maximal among all 2-splittable flows.

Given a network N' and a 2-splittable flow $\mathcal{F}'$, define the *fairness factor* of $\mathcal{F}'$ as

$$\rho(\mathcal{F}') \triangleq \frac{\min_i (f'_{i,1} + f'_{i,2})}{\max_i (f'_{i,1} + f'_{i,2})} \ .$$

Notice that $\rho(\mathcal{F}') \leq 1$. A 2-splittable flow $\mathcal{F}'$ achieving $\rho(\mathcal{F}') = 1$ is referred to as *fair*. We define the fairness of an instance N' as

$$\rho(N') \triangleq \max \left\{ \rho(\mathcal{F}') : f'(\mathcal{F}') \text{ is maximal} \right\} \ .$$

With the above definitions, we define several problems. The goal in the NODE 2-SPLITTABLE FLOW problem is to compute a maximum value 2-splittable flow, i.e., to find a 2-splittable flow $\mathcal{F}'$ that maximizes $f'(\mathcal{F}')$. The goal in FAIR 2-SPLITTABLE FLOW is to find a 2-splittable flow that maximizes ρ among the set of maximum value 2-splittable flows. The EDGE 2-SPLITTABLE FLOW problem is another variant of those problems in which the capacity function is on the edges, and the feasibility constraints are: $\sum_i \sum_{j:e' \in E(P'_{i,j})} f'_{i,j} \leq c'(e')$, for every $e' \in E$, where $E(P'_{i,j})$ are the edges along the path $P'_{i,j}$. All those problems may be defined in a straightforward matter for directed networks.

3 Hardness for General Capacities

In this section we show that NODE 2-SPLITTABLE FLOW is NP-hard and that it is NP-hard to approximate it within a factor of 1.5. It follows that even the feasibility question of FAIR 2-SPLITTABLE FLOW is NP-hard.

Our hardness results rely on the following reduction by Koch and Spenke [KS06].

Lemma 1 (Koch and Spenke [KS06])**.** *Given a a 3CNF formula φ, there is a polynomial time algorithm that constructs an instance of* EDGE 2-SPLITTABLE FLOW *with one source ($k = 1$) which satisfies the following properties:*

1. *$\varphi \in$ 3SAT implies that the optimal 2-splittable flow has value 3.*
2. *$\varphi \notin$ 3SAT implies that the optimal 2-splittable flow has value 2.*
3. *All edge capacities are either 1 or 2.*

First, we present a reduction from EDGE 2-SPLITTABLE FLOW with $k = 1$ to NODE 2-SPLITTABLE FLOW.

Theorem 1. *It is NP-hard to approximate* NODE 2-SPLITTABLE FLOW *within a ratio of 1.5, for any $k \geq 1$, even if $c'(v') \in \{1, 2, 3\}$, for every non-terminal node v'.*

Proof. Given an instance N'' of EDGE 2-SPLITTABLE FLOW with $k = 1$, we construct a NODE 2-SPLITTABLE FLOW instance N'. First, $S' = \{s'_1, s'_2, \ldots, s'_k\}$, where $s'_1 = s$, and $t' = t$. In addition, we split each edge $e'' = (v_{1,e''}, v_{2,e''})$ in N'' using a new vertex, denoted v'_e, whose capacity is $c'(v'_e) = c(e)$. Hence, $V' = V'' \cup \{v'_e : e \in E'\}$ and $E' = \left\{(v_{1,e''}, v'_e), (v'_e, v_{2,e''}) : e'' \in E''\right\}$. The capacity of the original vertices is 3, namely, $c'(v'') = 3$, for every $v'' \in V''$. It is straightforward to verify that there is a one to one correspondence between 2-splittable flows in N'' and in N'. The theorem follows from Lemma 1. □

Since NODE 2-SPLITTABLE FLOW is NP-hard, it follows that even finding a solution to FAIR 2-SPLITTABLE FLOW is NP-hard.

Corollary 2. *Finding a solution for* FAIR 2-SPLITTABLE FLOW *is NP-hard.*

The problem remains NP-hard even if one is given the optimum.

Theorem 3. *Given a* Fair 2-Splittable Flow *instance with the value of an optimal* 2*-splittable flow, it is NP-hard to find an optimal* 2*-splittable flow* $\mathcal{F}'$ *that maximizes* $\rho(\mathcal{F}')$*.*

Proof. The reduction from Theorem 1 implies that there is a polynomial time algorithm that given a 3CNF formula φ constructs an instance of Node 2-Splittable Flow with $k = 1$ that satisfies the following properties: (i) $\varphi \in$ 3SAT implies that the optimum is 3, (ii) $\varphi \notin$ 3SAT implies that the optimum is 2, and (iii) all node capacities are either 1 or 2. We extend the above construction. Denote the source and sink of the construction by s'_1 and t'_1. We add a second source s'_2. We add a new node u' and two edges (s'_2, u') and (u', t_1). We also add a new sink t and the edge (t'_1, t). The capacity of u is 4 and the capacity of t'_1 is 6.

By the above properties, there exists a 2-splittable flow form s'_1 to t'_1 whose value is 2. Moreover, the path from s'_2 to t'_1 can support four units of flow. Hence, the optimum is exactly 6. However, it is NP-hard to decide whether there exists a flow $\mathcal{F}''$ such that $\rho(\mathcal{F}'') = 1$. □

4 Node 2-Splittable Flow with Unit Capacities

In this section we show that Node 2-Splittable Flow with unit node capacities is solvable in polynomial time. We do so by using a reduction to Edge 2-Splittable Flow in directed networks with edge capacities of 1 or 2 units and employing known Maximum Flow results.

4.1 Using Maximum Flow

Before continuing, we invoke a couple of well known theorems for Maximum Flow in networks with directed edges and edge capacities (see, e.g., [AMO93, CCPS98, KT05]).

Theorem 4 (Flow Integrality)**.** *Given a* Maximum Flow *instance, if capacities are integral, then there exists an integral maximum flow. Furthermore, any augmenting path algorithm computes an integral flow.*

Theorem 5 (Flow Decomposition)**.** *Given a* Maximum Flow *instance, a flow* f *can be decomposed into the sum of at most* $O(|E|)$ *flows, each of which is a path flow or a cycle flow. Moreover, such a decomposition can be obtained in polynomial time. If* f *is integral, then the flow paths and cycles carry integral flows.*

Theorem 6 (Minimal s,t-cut)**.** *Given a* Maximum Flow *instance, there exists a minimum* s,t*-cut* $(S, \bar{S})$ *which is minimal with respect to set inclusion, namely such that* $S \subseteq S'$ *for any other* s,t*-cut* $(S', \bar{S}')$*. Moveover, this cut can be found in polynomial time.*

4.2 Reduction to Edge Capacities

Given an undirected network $N' = (G', c', S, t)$ with unit node capacities, we construct a new directed network with edge capacities $N'' = (G'', c, \{s, S\}, t)$ using standard techniques. The Network N'' is constructed from N' as follows :

1. A dummy source node s is added. A directed edge (s, s_i) with capacity $c(s, s_i) = 2$ is added from s to each source $s_i \in S$.

2. Each relaying node $v' \in V' \setminus (S \cup \{t\})$ is split into two nodes v_{in} and v_{out}, and is replaced by a directed edge $(v_{\text{in}}, v_{\text{out}})$ whose capacity is $c(v_{\text{in}}, v_{\text{out}}) = 1$.

3. Each edge (s_i, v') where $v' \in V' \setminus (S \cup \{t\})$ is converted into a directed edge (s_i, v_{in}) whose capacity is ∞.

4. Each edge (v', t) where $v' \in V' \setminus (S \cup \{t\})$ is converted into a directed edge (v_{out}, t), whose capacity is ∞.

5. Each edge (u', v') between two relaying nodes in N' is converted into two edges $(u_{\text{out}}, v_{\text{in}})$ and $(v_{\text{out}}, u_{\text{in}})$, whose capacities are ∞.

Edges in the N'' that replace relaying nodes in N' are referred to as *n-edges* and their capacity is 1. Edges in the N'' that replace edges in N' are referred to as *e-edges* and their capacity is infinite. Observe that the construction of N'' can be done efficiently.

Lemma 2. *Any minimum cut in N'' is composed of n-edges only.*

Proof: Follows from the facts that: (i) there are no direct edges between the sources in N' and the sink and (ii) the capacity of the e-edges is infinite. □

Denote by M the set composed of the outgoing nodes (v_{out}) of the edges in some minimum cut in N''. Such an s,t-cut can be computed in polyomial time (see Theorem 6). In the sequel it will be convenient to consider the subnetwork of N'' between the dummy source s and M.

Let N denote the directed network $N = (G, c, \{s, S\}, M)$, where G is the subgraph of G'' before and including M. A *flow path* in N is a pair (P, f), where P is a set of nodes along a simple path in G from s to some node in M and $f \in \mathbb{R}^+$.

A *2-valid flow* $\mathcal{F}$ in N is a set of flow paths in N such that there are at most two flows that traverse each source node s_i. A 2-valid flow is said to be *feasible* if it satisfies the edge capacities in N. Since all edges in M have unit capacity, a feasible 2-valid flow has all flows less than or equal to 1.

Lemma 3. *Consider a feasible 2-valid flow $\mathcal{F}$ in N. Let N'' be the network that has induced N. The flow paths in N can be continued all the way to the sink so that the resulting flow in N'' is feasible.*

Proof: The minimum cut M corresponds to a Maximum Flow in N''. The cut edges corresponding to M were saturated in this Maximum Flow problem in N'', namely all those edges carried a flow of 1. Feasibility of $\mathcal{F}$ includes the requirement that the capacity requirement is satisfied in the edges of M, namely the flow on all edges is at most 1. Thus continuation of the flows of $\mathcal{F}$ according to the original Maximum Flow maintains feasibility of the resulting flow in N''. □

Similarly to the definition of 2-valid flows in N, we define 2-valid flows in N''. A *2-valid flow* $\mathcal{F}''$ in N'' is a set of flow paths in N'' from s to t, such that there are at most two flows that traverse each source node s_i. A 2-valid flow is said to be *feasible* if it satisfies the edge capacities in N''.

Lemma 4. *There is a one-to-one correspondence between feasible 2-splittable flows $\mathcal{F}'$ in N' and feasible 2-valid flows $\mathcal{F}''$ in N''.*

Proof: First, a feasible 2-splittable flow $\mathcal{F}'$ in N' can be transformed in a straightforward manner into a feasible 2-valid flow $\mathcal{F}''$ in N''. Specifically, a path $P' = \langle s_i, v'_1, \ldots, v'_\ell, t\rangle$ in N' is transformed into the path $P'' = \langle s, s_i, v_{1,\text{in}}, v_{1,\text{out}}, \ldots, v_{\ell,\text{in}}, v_{\ell,\text{out}}, t\rangle$ in N''. To see that $\mathcal{F}''$ is feasible, notice that the flow through $(v_{\text{in}}, v_{\text{out}})$ is at most $c'(v) = c(v_{\text{in}}, v_{\text{out}}) = 1$. Moreover any flow through a path in N'' is at most 1 and each source s_i in N'' is traversed by at most two paths, thus the total flow over each edge (s, s_i) is at most 2.

Similarly, a feasible 2-valid flow $\mathcal{F}''$ in N'' can be transformed into a feasible 2-splittable flow $\mathcal{F}'$ in N'. Since the only outgoing edge from a node v_{in} is to v_{out}, a path P'' in N'' must have the structure $\langle s, s_i, v_{1,\text{in}}, v_{1,\text{out}}, \ldots, v_{\ell,\text{in}}, v_{\ell,\text{out}}, t\rangle$. Thus it can be replaced by $P' = \langle s_i, v'_1, \ldots, v'_\ell, t\rangle$, in N'. The resulting flow $\mathcal{F}'$ is feasible, since the capacity constraint of an edge $(v_{\text{in}}, v_{\text{out}})$ in N'' ensures that the capacity constraint of v is satisfied in N'. □

In the following sections, we proceed as follows. Starting with a network of type N' with unit node capacities, we derive the network of type N'' with 1,2 and ∞ edge capacities. A minimum cut in N'' induces the associated subnetwork of type N. We find a feasible maximal 2-valid flow in N, determine its fairness and invoke Lemma 3 to construct a feasible 2-valid flow in N''. The corresponding flow in N' (according to Lemma 4) provides the Routing Paths in N'. We use the notation $P' \propto P$ to show that the Routing Path P' in N' is obtained in this way from the flow path P in N.

In order to reduce the clutter in Figures (see e.g. Fig. 4), we depict networks of type N (and not N' or N''), with the following further simplifications: (i) the dummy source is not depicted; (ii) the n-edges of N are depicted as nodes; (iii) the e-edges of N are depicted as the corresponding undirected edges of the original N' .

4.3 Node 2-Splittable Flow for Unit Capacities

Consider a network with unit node capacities N', its equivalent network with edge capacities N'' and let F denote the value of maximal flow in N''.

Observation 5. $0 \leq F \leq 2k$*, where* k *denotes as before the number of sources.*

Proof. The capacity of the cut around s is $2k$. □

Theorem 7. Node 2-Splittable Flow *with unit capacities can be solved in polynomial time.*

Proof. Since capacities in N'' are integral, there exists a maximum flow $\mathcal{F}$ that is also integral, and such a flow can be found in polynomial time. Moreover, a decomposition of this flow into disjoint paths, each carrying an integral amount of flow, can be computed in polynomial time (see Section 4.1.) The edge capacities of N'' are 1, 2 or ∞, and therefore the decomposition consists of at most $2k$ paths, whose flow value is either 1 or 2. Moreover, since $c(s, s_i) = 2$ for every s_i, and all other capacities are 1, the flow decomposition induces a 2-splittable flow in the corresponding N'. □

5 Fair 2-Splittable Flow

In this section we study the problem of Fair 2-Splittable Flow and give algorithms for constructing optimal Routing Paths in networks with unit node capacities.

Consider a network with unit node capacities N', its equivalent network with edge capacities N'' and let F denote the value of maximal flow in N''. Recall that M denotes the set of outgoing nodes (v_{out}) of the edges in a minimum cut and N is the subnetwork of N'' between s and M. Since all edges in the cut have capacity 1, F is also the number of edges in the cut.

We shall need here an additional type of network. For a given source s_i, let $\hat{N}_i$ be the network N after the removal of all sources except s_i. The network $\hat{N}_i$ is a variant of N in which the there is only one edge with capacity 2 exiting s. Henceforth, we assume that the maximum flow in each $\hat{N}_i$ is at least 1, since otherwise there is no path from s_i to M and thus no path from s_i to the sink. Let $\hat{f}_i$ denote the maximum flow in $\hat{N}_i$, from s_i to M.

5.1 $F = 1$ and $F = 2$

Theorem 8. *If $F = 1$, there exists a fair 2-splittable flow in N' that can be computed in polynomial time. The Routing Paths in N' will be deducted from the flow paths in N.*

Proof. $F = 1$ implies $\hat{f}_i = 1$ for all s_i. Let $(\hat{P}_i, 1)$ be a flow path in $\hat{N}_i$.

The two routing paths $P'_{i,1}$ and $P'_{i,2}$ in N' are identical and are selected as $P'_{i,1} = P'_{i,2} \propto \hat{P}_i$. The flow values on these paths are selected $f'_{i,1} = f'_{i,2} = \frac{1}{2k}$. The resulting flow $\mathcal{F}'$ in N' is feasible, since the total load on a node is at most $2k \cdot \frac{1}{2k} = 1$. Moreover, since $f'(\mathcal{F}') = 1$, $\mathcal{F}'$ is optimal. Finally, the fairness factor is $\rho(\mathcal{F}') = 1$. □

Theorem 9. *If $F = 2$, it can be decided in polynomial time if a fair 2-splittable flow exists. If it exists, such a flow can be found in polynomial time. The Routing Paths in N' will be deducted from the flow paths in N.*

Proof. In the network N'' corresponding to N', change the capacity of the edges exiting s to $2/k$. Invoke an algorithm for Maximum Flow on this network. Since all capacities are multiples of $1/k$, the computed maximum flow can be viewed as consisting of multiples of $1/k$. (If k is even, then it consists of multiples of $2/k$.) Hence, the resulting Maximum Flow in N'' induces a 2-splittable flow $\mathcal{F}'$ in N' , where each of the paths has a value of $1/k$. Since $\rho(\mathcal{F}') = 1$ and $f'(\mathcal{F}') = 2$, it is optimal and fair. □

The flow $\mathcal{F}'$ may result in only one path from some sources to the sink, with flow $2/k$. As said in Section 1, we would like to split flows from sources into two paths. Next we define an algorithm for selecting such paths. Let v_1, v_2 be the nodes of M in N''. Since $F = 2$, the flow $\hat{f}_i = 1$ or 2 for all $s_i \in S$. The sources with $\hat{f}_i = 1$ can be of three types:

a) $\hat{f}_i = 1$ via v_1 only, like s_1 or s_2 in Fig. 1

b) $\hat{f}_i = 1$ via v_2 only, like s_5 in Fig. 1

c) $\hat{f}_i = 1$ via either v_1 or v_2, like s_4 in Fig. 1

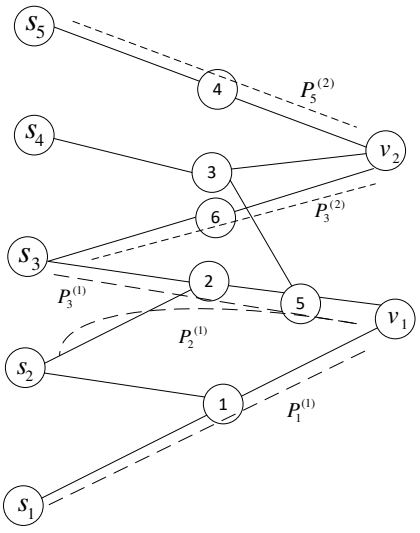

Figure 1: $F = 2$

Recall that to simplify the Figure, we draw it as described in the last paragraph of Section 4.2. Let

- S_1 be the set of sources s_i for which $\hat{f}_i = 1$ via v_1 and denote the corresponding path to v_1 by $P_i^{(1)}$
- S_2 be the set of sources s_i for which $\hat{f}_i = 1$ via v_2 and denote the corresponding path to v_2 by $P_i^{(2)}$
- S^b be the set of sources s_i for which $\hat{f}_i = 2$ and denote the corresponding (disjoint) paths to v_1 by $P_i^{(b1)}$ and to v_2 by $P_i^{(b2)}$.

Sources of type c) above are added to the smaller of the sets S_1, S_2, up to equal size. The remaining sources of type c), if any, are added in equal parts to the two sets. The size of the resulting sets will be denoted by k_1, k_2, k_b respectively. With these definitions we can design an easy algorithm for splitting traffic.

Theorem 10. *Suppose $F = 2$ and $k \geq 2$.*

Then:

a) if $|k_2 - k_1| \leq k_b$, then maximum throughput $f(\mathcal{F}') = 2$ and fairness $\rho(\mathcal{F}') = 1$ can be achieved.

b) if $k_2 > k_1 + k_b$, then fairness cannot be achieved. Maximum throughput $f(\mathcal{F}') = 2$ can be achieved with fairness factor $\rho(\mathcal{F}') = (k_1 + k_b)/k_2$

c) Similarly, if $k_1 > k_2 + k_b$

Proof. Note that *each path* of type $P_i^{(1)}$ is disjoint of *all paths* of type $P_i^{(2)}$. Thus the total flow through any network node is less than or equal to the total flow via v_1 and less than or equal to the total flow via v_2.

In case a), we show that maximum throughput $f(\mathcal{F}') = 2$ and fairness factor $\rho(\mathcal{F}') = 1$ are achieved with the following Routing Paths and associated flows:

- for $i \in S_1$, set $P'_{i,1} = P'_{i,2} \propto P_i^{(1)}$ with flow $f'_i = 2/k$
- for $i \in S_2$, set $P'_{i,1} = P'_{i,2} \propto P_i^{(2)}$ with flow $f'_i = 2/k$
- for $i \in S^b$, set $P'_{i,1} \propto P_i^{(b1)}$ and $P'_{i,2} \propto P_i^{(b2)}$ with flows $f'_{i,1} = (k_2 - k_1 + k_b)/(k \cdot k_b)$ and $f'_{i,2} = (k_1 - k_2 + k_b)/(k \cdot k_b)$ respectively.

With this selection, the flow in N' for each source is $f'_i = 2/k$ for all $s_i \in S$.

To show the above, it is sufficient to prove that for $s_i \in S^b$, holds $f'_i = 2/k$ and that the flow through v_1 and through v_2 is 1 . We have

$$\frac{k_2 - k_1 + k_b}{k \cdot k_b} + \frac{k_1 - k_2 + k_b}{k \cdot k_b} = 2/k$$

and

$$k_1 \cdot \frac{2}{k} + k_b \cdot \frac{k_2 - k_1 + k_b}{k \cdot k_b} = k_2 \cdot \frac{2}{k} + k_b \cdot \frac{k_1 - k_2 + k_b}{k \cdot k_b} = 1$$

In case b), $k_2 > k_1 + k_b$, , fairness cannot be achieved since any flow sent by sources in S^b via v_2 will disadvantage nodes in S_2. Maximum throughput is achieved with the following Routing Paths and associated flows:

- for $s_i \in S_1 \bigcup S^b$, set $P'_{i,1} = P'_{i,2} \propto P_i^{(1)}$ with flow $f'_i = 1/(k_1 + k_b)$
- for $s_i \in S_2$, set $P'_{i,1} = P'_{i,2} \propto P_i^{(2)}$ with flow $f'_i = 1/k_2$

The total throughput is $f(\mathcal{F}') = 2$ and the fairness factor is $\rho(\mathcal{F}') = (k_1 + k_b)/k_2$.

Similarly for c). □

5.2 Other Configurations

a) $F = 2k$: we have $2k$ disjoint paths from the sources to the sink, 2 from each source, each with flow 1. Those will provide the Routing Paths with corresponding flows $f_i = 2$ for all s_i for a total flow of $f(\mathcal{F}) = 2k$ and fairness factor $\rho(\mathcal{F}) = 1$.

b) $F = k$: The direct paths provide the optimum fair flows with $f_i = 1$ (see Fig. 2). Adjustments can be considered as in Section 7.

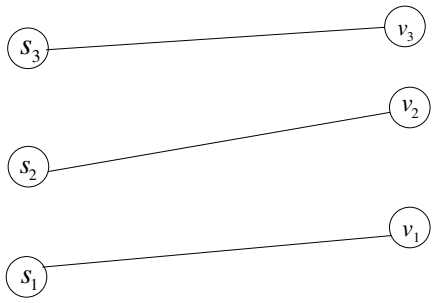

Figure 2: $F = k$

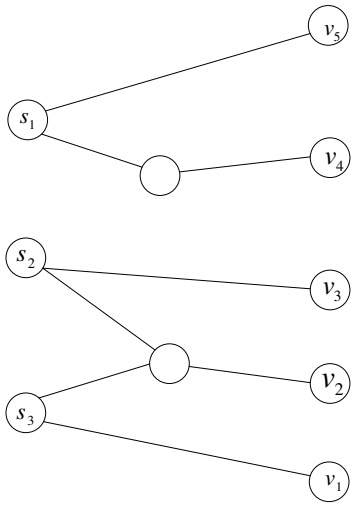

Figure 3: $F = (k + 2)$

c) $F \geq (k + 2)$: for example, $F = 5, k = 3$. The fair flows would be $f_1 = f_2 = f_3 = 5/3$. However, this is un-achievable with 2 paths from each source, as follows. For any source s_i holds $f_{i,1} \leq 1$ and thus for a flow from s_i of 5/3, its second path must carry $f_{i,2} \geq 2/3$. In order to have maximal total flow, the cut must be saturated, thus there must be another path, say $P_{j,1}$, with flow $f_{j,1} \leq 1/3$, that traverses the same cut edge as $P_{i,2}$. For a total flow of 5/3 from s_j, its second path must carry $f_{j,2} \geq 4/3$, which is a contradiction, since no path can carry more than 1 unit.

We can easily get $f_1 = 2$, $f_2 = f_3 = 1.5$ (Fig. 3) , thus $1 > \rho(N) \geq 2/3$.

d) $3 \leq F \leq (k - 1)$: for example $F = 3, k = 7$. The fair flow would be $f_i = 3/7$. There are networks where this cannot be achieved with 2 paths from each source. For example in Fig. 4, if the flow traversing each of the sources $s_1, \cdots, s_6$ is 3/7, there is no way to get a flow of 3/7 traversing source s_7 with only two paths. Maximum total flow (non-fair) can be achieved if we send 2/5 via each of the sources s_1, s_2, s_3, s_4, s_7 and 1/2 from each of the sources s_5, s_6.

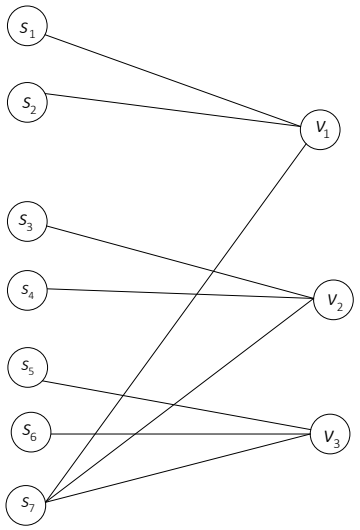

Figure 4: $F = 3, k = 7$

Thus $1 > \rho(N) \geq 4/5$.

6 Fair 2-Splittable Flow when $F = k + 1$

Consider a network with unit node capacities N', its equivalent network with edge capacities N'' and let F denote the value of maximal flow in N''. Recall that M denotes the set of outgoing nodes (v_{out}) of the edges in a minimum cut and N is the subnetwork of N'' between s and M. Since all edges in the cut have capacity 1, F is also the number of edges in the cut.

Consider the case when the size of the minimum cut F is one larger than the number of sources k. Since finding optimal fair routes for this case employs graph theoretic concepts and results in an especially interesting algorithm, we devote the present section to this type of configurations. With $F = k + 1$, a fair 2-splittable flow needs to give to each source a total flow of $(k + 1)/k$. We start with several preliminary observations and results.

A set of $\hat{k} < k$ sources is said to be *isolated* if the minimum cut of N'' with all other sources removed is of size $\hat{k}$. A Fair flow assignment is possible only if there are *no isolated sets of sources*. This is because if there is an isolated set of sources $\hat{S}$, the maximal flow of $F = k + 1$ in N'' from the dummy source s cannot give a flow of $f_i = (k + 1)/k$ to all $i \in \hat{S}$. A simple test to negate existence of isolated sets of sources is given by the following:

Lemma 6. *In N'', set $c(s, s_i) = 2$ and the capacity of all other edges exiting s to 1. Find the maximal flow from s to t. Repeat this separately for each source $s_i \in S$. If in all cases the maximal flow is $k + 1$, there are no isolated sets of sources.*

Proof. If there is an isolated set of sources, when selecting the link with capacity 2 as the link to any of those sources, we will get a maximal flow of k or less. □

Corollary 11. *If there are no isolated sets of sources, then for any source $s_i \in S$ holds $\hat{f}_i \geq 2$, namely there are at least 2 paths from i to the sink.*

Our Algorithm is based on the construction of Augmenting Paths in N. Because of the structure of N, flows and Augmenting Paths have special properties:

Lemma 7.

a) Any augmenting path is of value 1.

b) A flow can reach a (v_{in}, v_{out}) n-edge (representing a node v in the original graph G') only at the v_{in} end and if it does, it must traverse the n-edge.

c) An Augmenting Path can traverse only an empty forward edge or a backward edge with flow 1. *In the former case, it adds a flow of* 1 *to the edge. In the latter, it empties the edge.*

d) An Augmenting Path can enter an empty *(v_{in}, v_{out}) edge only in the v_{in} end, in which case it must traverse the n-edge and must leave over some edge outgoing from v_{out}.*

e) Consider a (v_{in}, v_{out}) edge that carries flow. *That flow is of size* 1. *Denote by (w_{out}, v_{in}) the adjacent edge the flow comes from and by (v_{out}, z_{in}) the adjacent outgoing edge the flow leaves over. An Augmenting Path may :*

- *reach the edge (v_{in}, v_{out}) at the v_{in} end on an incoming edge other than (w_{out}, v_{in}), in which case it must continue on the backward edge (v_{in}, w_{out})*
- *reach the edge (v_{in}, v_{out}) at the v_{out} end on the backward edge (z_{in}, v_{out}) and leave on an outgoing edge from v_{out} other than (v_{out}, z_{in})*
- *reach the edge (v_{in}, v_{out}) in the v_{out} end on the backward edge (z_{in}, v_{out}), continue on the backward edge (v_{out}, v_{in}) and then continue on the backward edge (v_{in}, w_{out})*

Proof. All properties follow directly from the facts that all (v_{in}, v_{out}) edges have unit capacities and except for (v_{in}, v_{out}), node v_{in} has only incoming edges and v_{out} has only outgoing edges. □

Case e) above results in the Augmenting Path "hijacking" the path of the flow it encounters and "transferring" the responsibility of continuing the Augmenting Path to the source traversed by that flow (illustration in Fig. 5 and exemplified below). That source will be referred to as the *Hijacked Source*.

The Hijacking process is illustrated in Fig. 5, where we use the shorthand L and R for v_{in} and v_{out} respectively. The edges $(L_2, R_2), (L_3, R_3), (L_6, R_6)$ are n-edges. We start with a flow of 1 on $s_2, L_6, R_6, L_3, R_3, L_2, R_2, n_2$. An Augmenting Path with flow = 1 arrives at R_2, traverses R_2, L_2, R_3, L_3, R_6 and leaves. While traversing the backward edges, it cancels the flow on those edges, thus we are left with a flow of 1 from s_3 to n_2 and a flow of 1 from s_2 via L_6, R_6, the latter continued on the Augmenting Path. Source s_2 is the *Hijacked Source*.

As seen presently, not all types of Augmenting Paths can be used in our Algorithm. We shall need Augmenting Paths with a special property, referred to as the *Self-Hijacking* property.

Definition: An Augmenting Path starting on edge (s, s_i) is said to be *Self-Hijacking* if either does not Hijack any source or the last Hijacked Source is s_i itself.

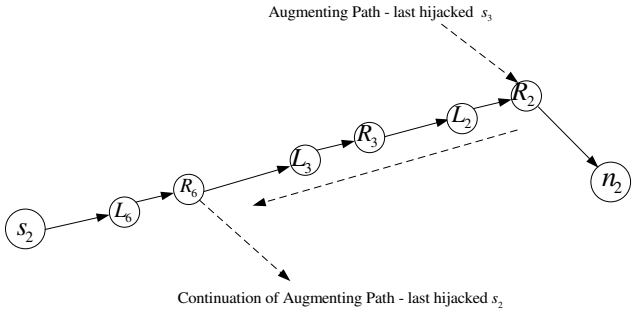

Figure 5: *Highjack*

6.1 Maximal Unfair Flow

In a graph N' with no isolated sets of sources, consider the corresponding networks N'' and N. Find the maximal flow in N'' from s to the sink. Maximal flow in a graph with uniform capacities of 1 and ∞ induces unity flows. Therefore in N there are $k+1$ paths from s to M, each with unity flow. Two of those paths in N traverse some source, s_1 say. All other paths traverse the other sources $s_2, \cdots, s_k$. Except for edge (s, s_1) that is common to two paths, all $k+1$ paths are disjoint.

Denote the following:

- M_1 is the set of the two nodes $\{v_0, v_1\}$ in M reached by the two paths that traverse s_1.
- $P_1^{(0)}, P_1^{(1)}$ are the paths in N starting at s, traversing s_1 and ending at M_1.
- $P_i^{(1)}$ is the path in N starting at s, traversing source s_i, $\ i = 2, \ldots, k$ and ending at M.
- V_1 is the set of nodes along the paths $P_1^{(0)}$ and $P_1^{(1)}$, not including the dummy source s.

We can obtain a maximum total flow, $F = (k+1)$, by selecting the Routing Paths $P'_{i,1}, P'_{i,2}$ in N' as follows:

- $P'_{1,1} \propto P_1^{(0)}; P'_{1,2} \propto P_1^{(1)}$ with flows $= 1$, thus $f_1 = 2$
- for $i = 2, \ldots, k$, select $P'_{i,1} \propto P_i^{(1)}$; $P'_{i,2} \propto P_i^{(1)}$ (namely $P'_{i,1} \equiv P'_{i,2}$) with flow $= 1$, thus $f_i = 1$

However, this choice induces an unfair assignment, whose $\rho = 1/2$. To obtain a fair assignment, we seek to find a second path, not necessarily disjoint from the paths selected above, for each of the sources $s_i, i = 2, \ldots, k$. The algorithm is specified in Section 6.2.

6.2 The Algorithm for selecting Routing Paths with fair assignment

In this subsection, we specify our Algorithm for selecting Routing Paths and prove that it provides *Maximal Fair Routing*. We first find a feasible 2-valid flow in N that also provides fairness. The selected Routing Paths are then obtained by the corresponding 2-splittable flow in N'.

The algorithm for obtaining the feasible fair 2-valid flow in N is performed in k stages. At the end of stage j, there are j sources with two paths to M and $k-j$ sources with one path to M. The configuration in Section 6.1 corresponds to the end of stage $j=1$. At each stage, the sources are renumbered, so that at the end of stage j, sources $1,\ldots,j$ have 2 paths to M and sources $j+1,\ldots,k$ have one such path.

For each $1 \leq j \leq k$, we denote:

- M_j is the set of cut edges reached by paths traversing sources $s_i,\ i \leq j$
- $P_{i,1}, P_{i,2}$ are the selected paths in N traversing source $s_i,\ i \leq j$
- $V_j = \bigcup_{i \leq j}(P_{i,1} \cup P_{i,2})$ (Recall that a path can be seen as a node set).

The Algorithm

a) The paths in N for s_1 are as in Section 6.1, $P_{1,1} = P_1^{(0)}; P_{1,2} = P_1^{(1)}$.

b) Send a flow of 1 on each of the paths $P_i^{(1)},\ i = 2,\ldots,k$ as defined in Sec. 6.1.

c) Using the Algorithm in Section 6.3, find a *Self-Hijacking Augmenting Path* from some source to V_1. Renumber the sources so that the found source is s_2. As shown in Section 6.3, the construction of the Self-Hijacking Augmenting Path results in:

   - 2 paths traversing s_2, disjoint except for (s, s_2), one terminating at a node in V_1 and one terminating at a node in M that is not in M_1. Denote this cut node by v_2 and $M_2 = M_1 \cup \{v_2\}$.
   - $(k-2)$ paths, one traversing each of $s_i,\ i = 3,\ldots,k$ and each terminating at one of the nodes in $M \backslash M_2$.

   Denote:

   - The path that traverses $s_i,\ i = 3,\ldots,k$ by $P_i^{(2)}$.
   - The path traversing s_2 and ending in V_1 by $P_u^{(2)}$ and the one ending in the cut node v_2 by $P_m^{(2)}$. Let w be the end node of $P_u^{(2)}$.

   The selected paths in N that traverse s_2 are selected as ($\sqcup$ denotes *concatenation*):

   - $P_{2,1} = P_u^{(2)} \sqcup$ the sub-path in V_1 from w to M
   - $P_{2,2} = P_m^{(2)}$

d) Proceed similarly for every $j > 2$: using the Algorithm in Section 6.3, find a *Self-Hijacking* Augmenting Path traversing some source to V_{j-1}. Renumber sources such that this source is s_j. Denote the path that traverses $s_i,\ i = (j+1),\ldots,k$ by $P_i^{(j)}$. Denote the 2 resulting paths

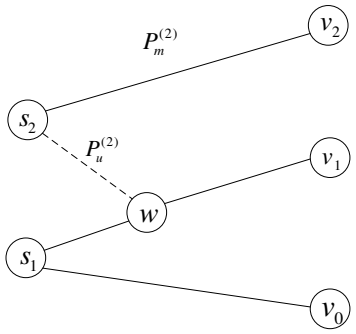

Figure 6: $k = 2$

traversing s_j by P_u^j and P_m^j. Let v_j denote the cut node that is the end of P_m^j. Let w be the end node of P_u^j.

The selected paths in N that traverse s_j are selected as:

- $P_{j,1} = P_u^j \sqcup$ the sub-path in V_{j-1} from w to M
- $P_{j,2} = P_m^j$

For future reference, we state a property in the following lemma, that follows directly from the construction in the Algorithm:

Lemma 8. *For every $j \leq k-1$, cut node v_j is reached only by selected paths $P_{j,2}$ and $P_{\ell,1}$, $\ell > j$.*

The construction is illustrated in Fig. 6, Fig. 7, Fig. 8 for 2,3 and 4 sources respectively.

Fig. 6 shows the construction for $k = 2$. Note that since we do not constrain the problem to planar graphs, the end of $P_u^{(2)}$ can be on either $P_{1,1}$ or $P_{1,2}$. The figure depicts the case when it is on $P_{1,2}$.

Fig. 7 shows the graph for $k = 3$. It assumes that it started with V_2 of Fig. 6, namely the end of $P_u^{(2)}$ was on $P_{1,2}$. It depicts all possible cases for $P_u^{(3)}$ (all the dashed lines). For the sequel, we take the case when the end of $P_u^{(3)}$ is $w^{(1)}$.

Fig. 8 shows the graph for $k = 4$, assuming that it started with the choice for V_3 as mentioned above. It depicts several possibilities (not all) for $P_u^{(4)}$.

Theorem 12. *-**Fairness:** Construct the Routing Paths as in the Algorithm above. There are flows $f_{i,1}, f_{i,2}$, $i = 1, \ldots, k$ in the final configuration V_k that provide a Fair Maximum Flow $\mathcal{F}$. Those flows can be calculated explicitly or found algorithmically.*

Proof. If a Fair Maximal Flow is achieved, the flows that traverse each source must be $f_i = (k+1)/k$ and the flows through each node in M must be 1. The numbering of the sources and the cut nodes is as defined in the Algorithm, namely in the order in which the selected paths have been found by the Algorithm.

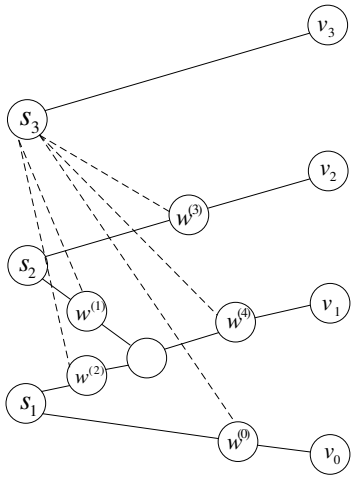

Figure 7: $k = 3$

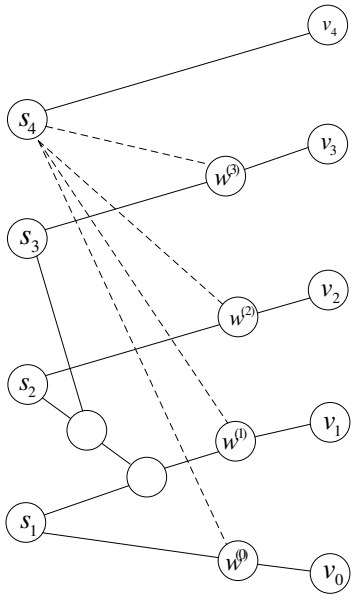

Figure 8: $k = 4$

We start with the last found source s_k (recall that there are k sources). We select $f_{k,2} = 1$, which is always feasible. Thus $f_{k,1} = 1/k$. We proceed to assign flows backwards to $s_{k-1}, \ldots, s_1$. We show by induction that for all $j < k$, fair flows $f_{j,1}, f_{j,2}$ can be calculated from previously assigned flows $\{f_{\ell,1}, f_{\ell,2},\ \ell \geq j+1\}$.

For any $j < k$, suppose we have already assigned fair flows to $s_{j+1}, ..., s_k$. We want to assign fair flows to s_j. Let $\hat{f}_j$ denote the total flow that traverses $s_{j+1}, ..., s_k$ minus the flow that enters the cut nodes in $M \setminus M_j = \{v_{j+1}, ..., v_k\}$. Then

$$\hat{f}_j = \frac{k+1}{k}(k-j) - (k-j) = \frac{k-j}{k} \leq 1 - \frac{1}{k}$$

Note that $\hat{f}_j$ is exactly the flow that traverses $s_{j+1}, ..., s_k$ and enters $v_0, v_1, ..., v_j$. Let $\hat{\hat{f}}_j$ denote the flow that traverses $s_{j+1}, ..., s_k$ and enters only v_j. The flow $\hat{\hat{f}}_j$ can be determined from the previously assigned flows $\{f_{\ell,1}, f_{\ell,2},\ \ell \geq j+1\}$. Obviously, $\hat{\hat{f}}_j \leq \hat{f}_j$. Thus $0 \leq \hat{\hat{f}}_j \leq 1 - 1/k$. But Lemma 8 implies that, in addition to $\hat{\hat{f}}_j$, only $f_{j,2}$ reaches v_j, thus $f_{j,2} = 1 - \hat{\hat{f}}_j$. Therefore $f_{j,2}$ can be calculated from $\{f_{\ell,1}, f_{\ell,2},\ \ell \geq j+1\}$ and has value $1/k \leq f_{j,2} \leq 1$. Thus it is feasible and its value is larger than or equal to $1/k$. This allows $f_{j,1} = (k+1)/k - f_{j,2}$ to be $f_{j,1} \leq 1$ and thus to be feasible and fair. □

We illustrate the above on several scenarios.

In Fig. 6, $k = 2$. The flows traversing each of the sources must be $3/2$. Flow $f_{2,2} = 1$, thus $f_{2,1} = 3/2 - 1 = 1/2$. Thus $f_{1,2} = 1 - 1/2 = 1/2$ and thus $f_{1,1} = 3/2 - 1/2 = 1$.

In Fig. 7, $k = 3$. The flows traversing each of the sources must be $4/3$. We assume that the paths traversing sources s_1, s_2 are as in Fig. 6. Consider the case when $w^{(1)}$ is the end node of $P_u^{(3)}$. The flows can be calculated as follows: $f_{3,2} = 1$, thus $f_{3,1} = 4/3 - 1 = 1/3$. Also, $f_{2,2} = 1$, thus $f_{2,1} = 4/3 - 1 = 1/3$. The flow entering v_1 must be 1, thus $f_{1,2} = 1 - f_{3,1} - f_{2,1} = 1/3$ and $f_{1,1} = 4/3 - 1/3 = 1$.

In Fig. 8, $k = 4$. The flows traversing each of the sources must be $5/4$. We assume that the paths traversing sources s_1, s_2, s_3 are as in Fig. 7.

We investigate several cases. Suppose $w^{(3)}$ is the end node of $P_u^{(4)}$. We have $f_{4,2} = 1$, thus $f_{4,1} = 5/4 - 1 = 1/4$. Thus $f_{3,2} = 1 - 1/4 = 3/4$, thus $f_{3,1} = 5/4 - 3/4 = 2/4$. Also, $f_{2,2} = 1$, thus $f_{2,1} = 5/4 - 1 = 1/4$. The flow over v_1 must be 1, thus $f_{1,2} = 1 - f_{3,1} - f_{2,1} = 1/4$, thus $f_{1,1} = 5/4 - 1/4 = 1$.

If $w^{(2)}$ is the end node of $P_u^{(4)}$, then $f_{4,2} = 1$, thus $f_{4,1} = 1/4$. Also, $f_{3,2} = 1$, thus $f_{3,1} = 1/4$. The flow $f_{2,2} = 1 - f_{4,1} = 3/4$, thus $f_{2,1} = 2/4$. Also $f_{1,2} = 1 - f_{2,1} - f_{3,1} = 1/4$, thus $f_{1,1} = 1$.

If $w^{(1)}$ is the end node of $P_u^{(4)}$, then $f_{4,2} = 1$, thus $f_{4,1} = 1/4$. Also, $f_{3,2} = 1$, thus $f_{3,1} = 1/4$. Also, $f_{2,2} = 1$, thus $f_{2,1} = 1/4$. Also, $f_{1,2} = 1 - f_{2,2} - f_{3,1} - f_{4,1} = 1/4$, thus $f_{1,1} = 1$.

If $w^{(0)}$ is the end node of $P_u^{(4)}$, then $f_{4,2} = 1$, thus $f_{4,1} = 1/4$. Also, $f_{3,2} = 1$, thus $f_{3,1} = 1/4$. Also, $f_{2,2} = 1$, thus $f_{2,1} = 1/4$. Also, $f_{1,2} = 1 - f_{2,1} - f_{3,1} = 2/4$, thus $f_{1,1} = 1 - f_{4,1} = 3/4$.

Given V_k obtained by our algorithm, the above flows can also be obtained algorithmically by running the non-linear optimization algorithm (in MATLAB say):

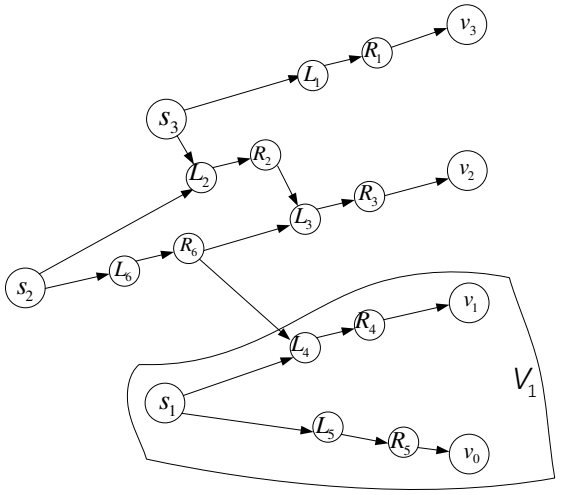

Figure 9: $Self - Hijacking$

$$\text{subject to } \begin{cases} \max \quad \sum_i \log(f_{i,1} + f_{i,2}) \\ \sum_i (f_{i,1} + f_{i,2}) = F \\ \text{capacity constraints on nodes of } M \end{cases}$$

Given that the maximum of the product of variables whose sum is given is reached when the variables are equal, the above optimization leads to fair flow.

6.3 Construction of Self-Hijacking Augmenting Paths

As seen above, step $(j+1)$ of the Algorithm is based on sending a flow of 1 over each path $P_i^{(j)},\ i = j, \ldots, k$ and finding a *Self-Hijacking Augmenting Path* traversing one of the sources $s_i,\ i = j, \ldots, k$. The construction of this Augmenting Path produces the second path for that source. The need for the *Self-Hijacking* property is shown in the following example:

Consider Fig. 9, where for the sake of clarity, we draw only some of the graph edges. We start with the paths (s, s_1, L_5, R_5, v_0) , (s, s_1, L_4, R_4, v_1), $(s, s_2, L_6, R_6, L_3, R_3, v_2)$, (s, s_3, L_1, R_1, v_3) and send a flow of 1 on the latter two (recall that the dummy source s is on the paths, but is not depicted in the Figures). The set V_1 is as depicted. We seek a self-Hijacking Augmenting Path traversing s_2 or s_3 to any node in V_1.

Selecting the Self-Hijacking-Augmenting Path $(s, s_2, L_2, R_2, L_3, R_6, L_4)$ from s_2 to V_1 results in two paths traversing s_2, one to v_2 and one to nodes of V_1. The first will be selected as $P_{2,1}$, while the second concatenated with the corresponding links in V_1 will compose $P_{2,2}$. This allows adding s_2 to the collection of sources for which we have found two paths. However, selecting instead the Augmenting Path $(s, s_3, L_2, R_2, L_3, R_6, L_4)$ from s_3 to V_1, which is not Self-Hijacking, results in two paths traversing s_3 towards v_2, v_3 and one path traversing s_2 to V_1. This outcome does not serve our algorithm.

Lemma 9. *At the end of stage j, $1 \leq j \leq k$ in the Algorithm of Section 6.2, there always exists an Augmenting Path* traversing each source $s_i,\ i = (j+1), \ldots, k$ *to some node in V_j.*

Proof. Note that at the end of stage j in the Algorithm, there is one path $P_i^{(j)}$, carrying 1 unit of flow, from each source $s_i,\ i = (j+1), \ldots, k$ to the cut. The total is $(k-j)$ units of flow. The set of sources $\{s_i,\ i = (j+1), \ldots, k\}$ contains $k-j$ sources and since there are no isolated sets of sources, the minimal cut of this set has at least $(k-j+1)$ nodes. Therefore, there is at least one additional path from each source to the sink. It must traverse V_j. □

Assume that we are at the end of stage j in the Algorithm of Section 6.2. That stage requires the construction of a *Self-Hijacking Augmenting Path*. Such a path is found as follows:

Find an Augmenting Path $\tilde{\tilde{P}}_i$ traversing each source $\{s_i,\ i = (j+1), \ldots, k\}$. If any of the Augmenting Paths is Self-Hijacking - done. Otherwise:

- Denote by $h(s_i)$ the last hijacked source of $\tilde{\tilde{P}}_i$.
- Find a series of sources $s^1, s^2, \ldots, s^\ell$, such that $s^m = h(s^{(m-1)}),\ m = 2, \ldots, \ell$ and $s^1 = h(s^\ell)$.
- Alter $\tilde{\tilde{P}}_i$ by deviating the path backwards all the way to $h(s_i)$ (for example in Fig. 9, continue the Augmenting Path from R_6 to L_6 and s_2). Denote the resulting Augmenting Path by $\tilde{P}_i$.
- Let $\tilde{P} = \tilde{P}_{s^1} \sqcup \tilde{P}_{s^2} \ldots \tilde{P}_{s^{(\ell-1)}} \sqcup \tilde{\tilde{P}}_{s^\ell}$
- Let $P_u^{(j+1)}$ be $\tilde{P}$ with all loops removed.

Theorem 13.

a) There exists a series of sources $s^1, s^2, \ldots, s^\ell$, as above.

b) The path $P_u^{(j+1)}$ constructed above is a Self-Hijacking Augmenting Path originating at s^1 and terminating at some node in V_j.

Proof. Part a) follows from the fact that there is a finite number of sources. To prove part b), note that since we are following Augmentation Paths in graphs with unity capacities, the only way for $P_u^{(j+1)}$ to be blocked is if it attempts to employ an edge that $P_u^{(j+1)}$ itself had employed before. In other words, that edge was visited before by $P_u^{(j+1)}$. This contradicts the fact that $P_u^{(j+1)}$ is loop free. Finally, note that $h(s^\ell) = s^1$, namely the last hijacked source of $\tilde{\tilde{P}}_{s^\ell}$ is s^1, thus $P_u^{(j+1)}$ is a Self-Hijacking Augmenting Path. Moreover, $\tilde{\tilde{P}}_{s^\ell}$ terminates at some node in V_j and therefore so does $P_u^{(j+1)}$. □

7 Adjustments to the Basic Algorithms

If $F = 1$, we can improve the throughput of HDR, if the two Routing Paths are as disjoint as possible [Seg15]. The algorithm for making them as disjoint as possible is:

1. Find all 1-cuts in the network. This is easily done by finding any 1-cut with a Max Flow algorithm, setting its harvesting rate to 2 and repeating the process.

2. For all 1-cuts along the shortest path, set their harvesting rate to 2.

3. Now the Max Flow will be at least 2 and will provide two disjoint paths, each with flow of 1.

4. Translating these paths to Routing Paths, we obtain two paths P^A and P^B, each with flow 1/2, as disjoint as possible. The throughput will be $T = 1$.

Consider the network of Fig. 10. The Harvesting Rate of all relaying nodes corresponds to 10 *packets/slot*. The nominal maximum flow is 10 packets/slot. It can be achieved if both paths are that can be obtained with both paths being $1 - 2 - 4 - 5$. With the algorithm above, we get one path $1 - 2 - 4 - 5$ and the other $1 - 3 - 4 - 5$. Simulation shows that HDR gets throughput of In the first case, we get an throughput of $\mathcal{F} = 9.13$ packets/slot, while in the second case $T = 9.81$ packets/slot. Note that the nominal throughput of 10 packets/slot cannot be achieved, since as explained in the Appendix, the simulation takes into account energy spent by nodes for control packets and for overhearing headers of data packets not intended for the node itself.

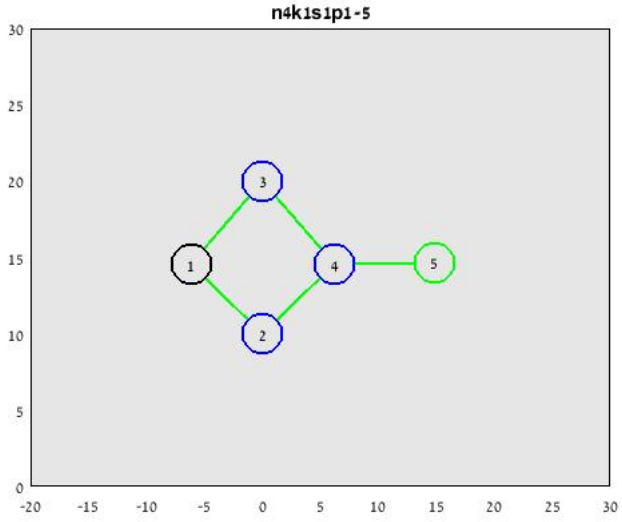

Figure 10: One Source - Three Relaying Nodes

In all configurations, the common portion of two paths can be translated into 2 paths that are as disjoint as possible as shown above.

References

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[Bai03] Georg Baier. *Flows with Path Restrictions.* PhD thesis, TU Berlin, 2003.

[BKS05] Georg Baier, Ekkehard Köhler, and Martin Skutella. The k-splittable flow problem. *Algorithmica*, 42(3-4):231–248, 2005.

[CCPS98] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[ELSY17] S. Erlich, A. Lavzin, A. Segall, and M.I. Yomtovian. A simulation system for energy harvesting networked tags (EnHANTs). Technical Report arXiv:1701.01799 [cs.NI], 2017.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[KS06] Ronald Koch and Ines Spenke. Complexity and approximability of k-splittable flows. *Theoretical Computer Science*, 369(1-3):338–347, 2006.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[LS] A. Lavzin and A. Segall. Hdr - a routing algorithm for multiple source networks with energy harvesting units. In *submitted to MEDHOCNET 2018*.

[MGS+15] R. Margolies, M. Gorlatova, J. Sarik, G. Stanje, J. Zhu, Miller P., M. Szczodrak, B. Vigraham, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman. Energy harvesting active networked tags (enhants): Prototyping and experimentation. In *ACM Transactions on Sensor Networks*, volume 11, pages 62–99, 2015.

[PR99] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *WMC SA'99, 2nd Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb 1999.

[Seg15] A. Segall. HDR - a hysteresis routing algorithm for energy harvesting tag networks. In *MEDHOCNET 2015, Algrave, Portugal*, June 2015.