Non-Bifurcated Routing in Wireless Multi-Hop Mesh Networks

Abdullah-Al Mahmood, Ehab S. Elmallah
Department of Computing Science
University of Alberta
Edmonton, T6G 2H1, Canada
E-Mail: {amahmood,ehab}@cs.ualberta.ca

Ahmed Kamal
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
E-mail: kamal@iastate.edu

***Abstract*—In this paper we consider traffic routing in 802.11-based multi-hop Wireless Mesh Networks (WMNs). Interest in such networks arise since they offer flexible, and cost effective means of providing Internet connectivity to communities of subscribers. Successful deployment of such networks, however, hinges on the ability of the network to serve subscribers at the data rates specified by service agreements, as well as providing quality of service to certain key traffic types, such as TCP traffic, delay-jitter sensitive traffic, and traffic that requires synchronized delivery to end users. Since delays on different routes in such networks may vary widely, routing of the above traffic types can potentially benefit from non-bifurcated routing schemes that do not split flows among multiple paths. In this paper, we formalize the problem of non-bifurcated routing, while meeting subscriber demands, as an optimization problem. We present a heuristic algorithm that utilizes results from the theory of maximum flows, and insights into the routing problem to obtain efficient solutions. Simulation experiments indicate improved achieved throughput, and delay-jitter results over the use of the standard Dynamic Source Routing (DSR) algorithm.**

Keywords: wireless mesh networks, fixed broadband wireless access networks, non-bifurcated routing, flow algorithms

I. Introduction

In recent years, wireless broadband access (WBA) networks that support multimedia services have received increased attention as a low cost, and flexible means that enable network providers to extend Internet connectivity to subscribers. Examples of applications of such networks include serving subscribers in rural areas, as well as subscribers in city-wide hot-spots. The IEEE WirelessMAN/HUMAN 802.16 [1] family of standards aim at enabling rapid, and inter-operable worldwide deployment of such networks. In the standard, a mesh base station (Mesh BS) is a node that provides connectivity to backhaul services outside the mesh network, and a mesh subscriber station (Mesh SS) is a node that provides connectivity between subscriber equipment and a base station. The standard defines two modes of operation of such networks: the *point-to-multipoint* (PMP) mode where all communications are controlled by a single Mesh BS, and the *mesh* mode where traffic can be relayed in a multi-hop fashion between subscriber station's and one, or more, base stations.

Multi-hop networks utilizing the mesh mode minimizes equipment cost by utilizing subscriber stations to forward traffic to the base stations. Single-radio multi-hop mesh networks appears to be the most successful commercial architecture thus far.

In this paper, we consider traffic routing in 802.11-based multi-hop wireless mesh networks (WMNs) of the type mentioned above, where subscriber units connect to wireless mesh routers (nodes), and routers collaborate in forwarding the traffic to one, or more, gateway (Mesh BS) .

Resource management algorithms for such WMNs are required to allocate bandwidth to subscribers in a controlled manner, so as to satisfy service agreements. In addition, such mechanisms are required to provide acceptable throughput for key traffic types, such as TCP traffic, delay-jitter sensitive traffic, and multimedia traffic that requires synchronized delivery to end users. Since delays on different routes in such networks may vary widely, routing of the above traffic types can potentially benefit from non-bifurcated routing schemes that do not split flows among multiple paths. The issue of avoiding traffic splitting has been mentioned (but not dealt with), for example, in [2].

A novel aspect of our work here is the development and evaluation of a non-bifurcated routing algorithm. It is shown that the performance of the resulting algorithm is competitive when compared with the well known dynamic source routing (DSR) algorithm.

For prior work, we note that [3] presents a survey of many key architectural aspects of WMNs, and discusses a number of research challenges in the area. Routing algorithms for WMNs, in general, build on the current knowledge, and experience gained from the analysis of ad hoc routing algorithms. Recent research work in the area of 802.11-based multi-hop WMNs concern the development of methodologies to characterize the capacity of multi-channel networks, and the development of centralized, and distributed routing, and channel assignment algorithms to manage the available bandwidth. Examples of recent work in the above directions include the work of [2], [4]–[6]. We note, however, that none of the above work provides a solution to the non-bifurcated routing problem considered in the paper.

The work of [2] aims at characterizing the capacity region of a synchronized (time-slotted) multi-channel WMN, where each node may have fewer RF interfaces than the available

orthogonal channels. Among other results, [2] provides necessary conditions for testing the feasibility of data rate vectors in such networks, and use the obtained necessary conditions to derive upper bounds on the achievable throughput. Similarly, the work of [4] considers routing in slotted multi-channel WMNs, and presents algorithms for joint channel assignment, and routing for throughput maximization.

We note that, in contrast to our work here, the network model used in [2], [4] assumes synchronized operation of all routers in the network, where data transmission occurs during well defined time slots.

In [5], [6], the authors present distributed channel assignment, and routing algorithms for 802.11-based multi-hop multi-channel WMNs. The network model adopted in the above work does not assume synchronized operation of the routers. The algorithm presented in [5] approaches the problem of minimizing the effect of the MAC layer contention by periodically exchanging measured channel usage information between a node and its neighbours, and implelemnting a scheme that utilizes such information to change the RF interface-to-channel mappings at each node. On the other hand, the work of [6] does not explicitly consider the impact of contention due to traffic from nearby nodes. Rather, the implemented routing algorithm (a source-routed link-state protocol derived from DSR) takes into consideration the measured packet loss rate due to collisions.

In contrast with the above measurement based approaches, our proposed framework assumes that traffic demands from each node are sent periodically to a central node that is responsible for computing a new set of routes, and subsequently distributing the computed routes to all other nodes; the central node computes the routes by applying our devised algorithm that takes channel contention into consideration.

The rest of the paper is organized as follows. Section 2 formulates the non-bifuracted routing problem as a network flow problem under suitable constraints. Sections 3 describes a core function that is utilized in an iterative way by the devised algorithm to solve the routing problem. Section 4 discusses possible extensions of the function to handle multi-channel WMNs. Performance results are then presented in Section 4, followed by concluding remarks.

II. Problem Formulation

A. System Model.

Throughout the paper, we consider a WMN consisting of fixed wireless routers (the WMN nodes) that are capable of aggregating traffic from suscriber units. Some routers act as gateways to the wired Internet. Routers utilize multi-hopping to relay subscriber traffic to (or form) the gateway(s). In the general case, each node is equipped with a number of 802.11-based RF interfaces, and there is a number of orthogonal wireless channels to utilize.

To model the interference, and conetntion of flows in the 802.11-based RTS-CTS-DATA-ACK environment, we adopt the following commonly used assumptions (see, e.g., [2], [5], [7]):

1) All nodes are assumed to have the same transmission range, denoted R_T, and interference range, denoted R_I, where $R_I \geq R_T$,
2) Two nodes that are within the R_T range of each other can establish direct communication.
3) If two nodes are not within the R_T range, but within the interference range (R_I) of each other, they cannot communicate directly, however, they interfere with each other.
4) Two flows that use orthogonal channels do not interfere with each other.
5) As adopted in [2], using the protocol interference model of [8], a transmission on a certain link and channel is successful when all potential interferers in the neighbourhood of the sender and the receiver are silent on the channel for the duration of the transmission.

Thus, as remarked in [7], under the above assumptions neighbourhood and flow contention are commutative properties. In addition, the above asumptions implies that two flows contend with each other if either the sender, or the receiver of one flow coincides with, or lies within the interference range of the sender, or the receiver of the other flow.

For the purpose of providing non-bifurcated routing, we define a flow as a sequence of packets that can be uniquely identified at each WMN node. Each flow is required to be routed to (or form) one of the available gateways without splitting at any hop on the route. In addition, each flow requires a certain amount of data rate. We assume that the network operator defines the equivalent data rate of a flow unit, and that applications make requests to their serving WMN nodes in integer multiples of such units.

B. Single Channel Problem Formulation

In this section, we consider traffic routing over a single wireless channel operating under the interference model mentioned above. In particular we formulate the problem of maximizing the total amount of flow (throughput) served by the network at any time as a network flow problem. Section 3 develops a core function to search for a possible way to increase the flow in a given network, and section 4 discusses possible extensions of the function to handle multi-channel WMNs. The effectiveness of the given problem formulation, and the devised solution is investigated by simulation in section 5.

To start, we denote by $G = (V, E_T, E_I)$ the directed graph on the set V of WMN nodes, of which a subset of nodes $GW \in V$ serve as gateways. E_T denotes the set of transmission edges, and E_I is the set of interference edges. As noted above, since neighbourhood is assumed to be commutative, if a directed edge $(x, y) \in E_T$ (or E_I) then the reverse edge $(y, x) \in E_T$ (respectively, E_I). Moreover, by the above remarks, a flow $f(x, y)$ affects the network in the same way as a flow $f(y, x)$ of equal amount on the reverse edge. Hence, a flow on a route from a mesh node x to a gateway g affects the network in the same way as a flow of equal amount on the same route from g to x.

Thus, with out loss of generality, we may asume that all flow demands are directed towards the gateway(s). Moroever, since flow demands from a gateway's own subscriber units are routed directly to outside the mesh, we simplify the problem by omitting such demands from the problem formulation.

We formulate the throughput maximization problem using the following additional notation.

- $D(x)$ (requested flow demands at node x): a vector $(d_i(r)|\ i = 1, 2, \cdots, |D(r)|)$, where the ith requested flow demand has value d_i units. The vectors of requested flow demands are assumed to be sent periodically to a central node that is responsible for computing new sets of routes, and subsequently distributing the computed routes to all other nodes.
- $S(x)$ (accepted flow demands at node x): a vector specifying the flow demands of $D(x)$ that are selected for routing.
- f: a vector (computed by an algorithm) specifying for each transmission edge (x, y) a flow of value $f(x, y)$.
- $f(X, Y)$ (aggregate flow notation): for two sets of nodes $X, Y \subseteq V$, the sum of flows assigned to transmission edges, where each edge has its tail in X and its head in Y. That is, $f(X, Y) = \sum \{f(i, j)|\ i \in X, j \in Y,$ and $(i, j) \in E_T\}$.
  We also write $f(X, u)$ (or, $f(u, X)$) if one set is just a single node u.
- $E_T^{int}(x)$: denotes the set of transmission edges where each edge has at least one of its end nodes located within the interference range of node x.
- $f(D(x))$, $f(S(x))$, $f(E_T^{int}(x))$: $f(D(x))$ (or, $f(S(x))$ denotes the sum of all demands in the vector $D(x)$ (respectively, $S(x)$). $f(E_T^{int}(x))$ denotes the sum of all flows assigned to transmission edges in $E_T^{int}(x)$.
- $C(x)$: the available channel capacity at node x. The model allows different nodes to have different channel capacities to account for the possible outside interference on wireless channels in the unlicensed wireless bands.

To simplify the presentation, we also define the channel *loading factor* at node x, denoted $\ell(x)$, caused by a given flow vector f, as:

$$\ell(x) = f(V, x) + f(x, V) + f(E_T^{int}(x))$$

where $f(V, x)$ is the sum of all flows entering x, $f(x, V)$ is the sum of all flows leaving x, and $f(E_T^{int}(x))$ is the sum of all interfering flows at x. We note that, $f(S(r))$ (the sum of all flows entering x from its own subscriber units, and accepted for routing) does not contribute to the channel loading factor, since we assume that such flows do not use the same wireless channel used for backhaul communication between the WMN nodes.

Our model hypthesizes that the vectors $(S(x)|\ x$ is a non-gateway node$)$ of accepted traffic flows admit non-bifuracted routing to the gateway(s) if there exists a flow vector f that satisfies the following constraints:

Channel Capacity Constraint. For any node x, the channel loading factor caused by the flow vector f does not exceed the available channel capacity at the node: that is, $\ell(x) \leq C(x)$.

Flow Conservation Constraint. For any non-gateway node x, the sum of the outgoing flows from x equals the sum of the incoming flows into x, plus the flow demands accepted for routing; that is, $f(x, V)) = f(V, x) + f(S(x))$.

Flow indivisibility Constraint. For any anode x, and flow demand $d_i(x) \in D(x)$, a flow of amount $d_i(x)$ is assigned a route from x to a gateway in G.

Finally, the throughput maximization problem is to maximize the total flow routed to the gateway(s). That is, using the aggregate flow notation, we want to maiximize $f(V, GW)$.

C. Background Results and Remarks

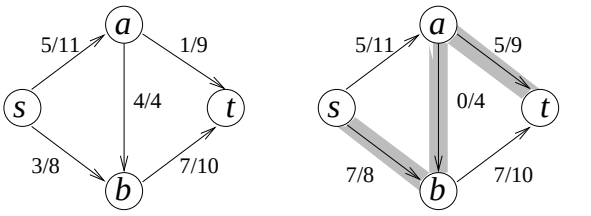
A few remarks about the computational complexity of the above throughput maximization problem are now in order.

First, the throughput maximization problem with arbitrary integer requested flow demands (i.e., the numbers in a D vector), can be shown to be NP-complete. In this general case, the PARTITION problem ([SP12] in [9]) can be transformed into the above problem in polynomial time.

Second, we note that the simplified throughput maximization problem where all terms of the form $f(E_T^{int}(x)) = 0$ (i.e., ignored), and all demand vectors include unit flows only, is equiavlent to a maximum flow problem with multiple sources and sinks, and capacity constraints on nodes. This latter problem, however, can be solved using an algroithm for solving the standard 2-terminal maximum flow problem (see, e.g. [10]).

Third, a prominent class of algorithms for solving the 2-terminal maximum flow problem (e.g. see [10] for the Ford-Fulkerson, and the Edmonds-Karp algorithms), relies on the idea of repeatedly finding a *flow augmentation* path (FAP), until no such FAP exists. A FAP is a sequence of edges that form an undirected path from a source node s to a terminal node t. Thus, the path may traverse some edges in the forward direction, and traverse other edges in the reverse direction. So, relative to such an undirected path P, some edges are *forward* edges, and some edges are *reverse* edges. It is known that if each forward edge admits a flow increase by v units, and each reverse edge admits a decrease of its current asigned flow by v units, then the adopting such flow changes along an (s, t) FAP results in a net increase of the total flow from s to t by v units. The following example illustrates the above well known concept.

Example. Figure 1(a) illustrates an instance of the maximum (s, t)-flow problem, where 8 units of flow are sent from the source s to the terminal t. The undirected path $P = (s, b, a, t)$ is a FAP. P traverses the two edges (s, b) and (a, t) in the forward direction, and the edge (a, b) in the reverse direction. Increasing the flow along each forward edge by 4 units, and



(a) (b)

Fig. 1: Flows in a directed network (x/y denotes a flow of value x on an edge of capacity y)

decreasing the flow along each reverse edge by 4 units, give the set of flows in Fig. 1(b) of net value $= 12$ units. ■

III. Single Channel Flow Augmentation Algorithm

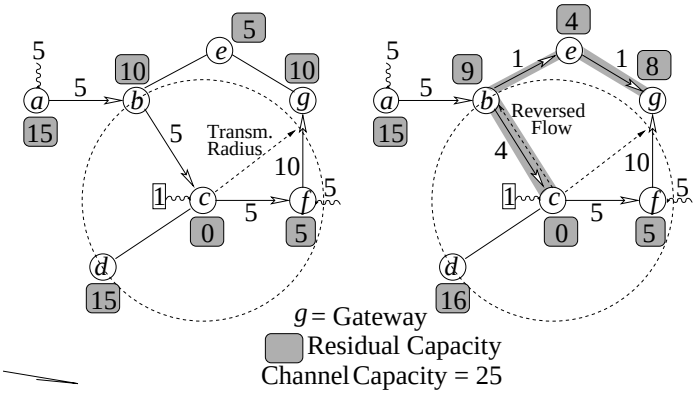
Building on the strength of the flow augmenting path (FAP) concept in solving the classical two-terminal maximum flow problem, in this section we extend the idea to work in the present context of non-bifurcated routing in WMNs. We call the new type of paths, *interference-constraint* FAPs (or, IC-FAPs for short). Two basic issues related to the formulation of IC-FAPs are: how to define such structures? And, how to find such IC-FAPs efficiently (if possible)?

The basic observations presented below are used to design a heuristic search algorithm, called ICFAP_Find. To simplify the presentation, we henceforth assume that the transmission radius (R_T) equals the interference radius (R_I); the simplification does not restrict the generality of the developed methodologies.

A. IC-FAPs Definition and Observations

Given the connectivity graph $G = (V, E)$ of a WMN, with a vector of flow values assigned to the edges, we define an IC-FAP of value v from some demand node x to a network's gateway as an undirected path such that increasing the flow value on each forward edge by v units, and decreasing the flow value on each reverse edge by v units yield a flow vector that does not violate the channel capacity constraints at any node. The following example illustrates the above definition.

Example. Fig. 2(a) illustrates a WMN where node g is a gateway. Initially, the available channel capacity at each node is assumed to be $C = 25$ units. Fig. 2(a) illustrates a flow of 5 units sent along the path (a, b, c, f, g), and another flow of 5 units sent along the edge (f, g). The total flow into the gateway is 10 units. The available residual channel capacity at each node appears inside an adjacent square. For example, the load factor at node f, $\ell(f) = f(c, f) + f(f, g) + f(b, c) = 20$ units, where the first two terms are pass-through flows, and the third term is an interference flow. Thus, node f's residual capacity $C(f) = 25 - 20 = 5$ units. Likewise, the load factor at node c, $\ell(c) = f(b, c) + f(c, f) + f(a, b) + f(f, g) = 25$ units, where the first two terms account for the pass-through flows, and the last two terms account for interference flows. Thus, node c's residual capacity $C(c) = 25 - 25 = 0$ unit.



(a) (b)

Fig. 2: An example of an IC-FAP in a wireless mesh network.

The network in Fig. 2(a) admits the IC-FAP of value $v = 1$, highlighted in Fig. 2(b). The IC-FAP is from node c to the gateway g along the path $P = (c, b, e, g)$. Here, the first edge (b, c) is traversed in the reverse direction (its associated flow is decreased by $v = 1$ unit), and the two edges (b, e) and (e, g) are traversed in the forward direction (their flows are increased by $v = 1$ unit each). Fig. 2(b) also illustrates the resulting residual capacities at each node after modifying the flows along the indicated IC-FAP.

■

A few remarks now follow in order. First, we remark that, as in the case of the standard maximum flow FAPs, starting with a feasible flow vector, and making flow changes along an IC-FAP yield a new flow vector that satisfies the flow conservation constraints, as the original flow vector. Thus, starting with the zero flow vector, and repeatedly finding IC-FAPs can be used to obtain feasible flows with higher net flow amounts into the gateway.

Second, given a network and an initial flow vector, the ability to employ the above mentioned iterative scheme to solve the throughput maximization problem hinges on the ability to find an IC-FAP efficiently. Currently, no such efficient exact algorithm appears to be known. Finding such an IC-FAP from some demand node s to a gateway t appears to face the following computational difficulty: if x is a possible intermediate node, $s \neq x \neq t$, on a sought after IC-FAP, then the ability to extend a given path segment from s to x, so as to reach t, appears to depend on the exact distribution of loading factors caused by increasing the flows along the forward edgess, and decreasing the flows along the reverse edgs of the given segment. Thus, an exact algorithm may have to examine exponentially many paths from s to x. Such apparent computational difficulty does not exist in the search problem for finding a FAP in the standard two-terminal maximum flow problem.

Third, in a search for an IC-FAP, if $P = (x_0, x_1, \cdots, x_r)$, $r \geq 2$, is a directed path from some node x_0 to another node x_r along which a unit of flow can be sent by traversing each

edge in the forward direction, and if (x_0, x_r) is also an edge in G, then the search algorithm should not consider the longer path P. To see why, let y be any arbitrary node in G, and denote by $\ell(y)$, and $\ell'(y)$ the loading factors that result from sending a unit flow along the edge (x_0, x_r), and the path P, respectively. One may then verify that $\ell(y) \leq \ell'(y)$, and hence using the edge (x_0, x_r) is always the better choice. The algorithm presented below uses the above remark by giving preference to extending a path to reach nodes that are as close as possible to the gateway.

B. IC-FAP Search Algorithm

In this section we present a search algorithm, called ICFAP_Find (cf., Fig. 4) for the flow augmentation problem mentioned above. Table I describes the function inputs and output.

TABLE I: Function ICFAP_Find inputs and output.

Input Parameters:	
G:	The directed graph of a WMN
$flow$:	An array specifying for each directed edge (x, y) a flow value $flow(x, y)$; the values constitute a feasible flow in the network G
cap:	A vector specifying for each node x the residual channel capacity obtained by taking all $flow$ values into consideration
$Nforward$:	An array specifying for each node x two closest neighbours of x to the gateway t, $x \neq t$; if x has one neighbour, the second node is set to null (zero value)
v_{req}:	The required flow increment value of the sought after IC-FAP from demand node s to gateway t
s:	A node with a required unsplittable flow demand of value v_{req}
t:	A target gateway in G
Output:	
P	an IC-FAP from s to t of the required value v_{req}, returned upon a successful search (else, the returned path P is empty)

As can bee seen, the function takes as input the connectivity graph G of a WMN, an array $flow$ of current flows routed in the network, and the resulting residual channel capacity at each node. The function searches for an IC-FAP from a given demand node s to the network's gateway t that increases the net flow in the network by the amount specified by v_{req}.

The input array $Nforward$ is one ingredient in a mechanism utilized by the function to bound the number of stored IC-FAPs from s to any intermediate node x during the search. Specifically, if $Nforward[x] = (y_1, y_2)$ (or, $(y_1, 0)$ if x has only one neighbour), then y_1 and y_2 are closest neighbours of x to the gateway t; the algorithm keeps a collection of IC-FAPs from s to x, where each IC-FAP induces a certain distribution of channel loading factors at these two distinguished nodes y_1 and y_2. Two different partial IC-FAPs from s to x that result in the same distribution of channel loading factors at y_1 and y_2 are then considered indifferent by the function. Hence, only one of the two paths is kept in the stored collection.

A second ingredient in bounding the number of examined IC-FAPs from s to x is a table, denoted T_x, maintained for each node x. T_x provides a key-value mapping from pairs of integers (loading factors at the $Nforward[x] = (y_1, y_2)$ nodes), to IC-FAPs from s to x; the net flow along each of the stored IC-FAPs is v_{req} units.

T_x is used in the following way. Initially, T_x is empty (no key exists in the domain of T_x). Subsequently, if $Nforward[x] = (y_1, y_2)$, and $(\ell(y_1), \ell(y_2))$ is a key in T_x (i.e., a pair of channel loading factors at nodes y_1, and y_2, respectively), then $T_x[\ell(y_1), \ell(y_2)]$ is an IC-FAP, denoted P, from s to x; we interpret that the input vector $flow$, combined with the flows assigned to the IC-FAP segment P result in channel loading factors of values $\ell(y_1)$, and $\ell(y_2)$, at nodes y_1 and y_2, respectively. Moreover, if $y_2 = 0$ (the null value), then $\ell(y_2) = 0$. The following example illustrates the above concepts.

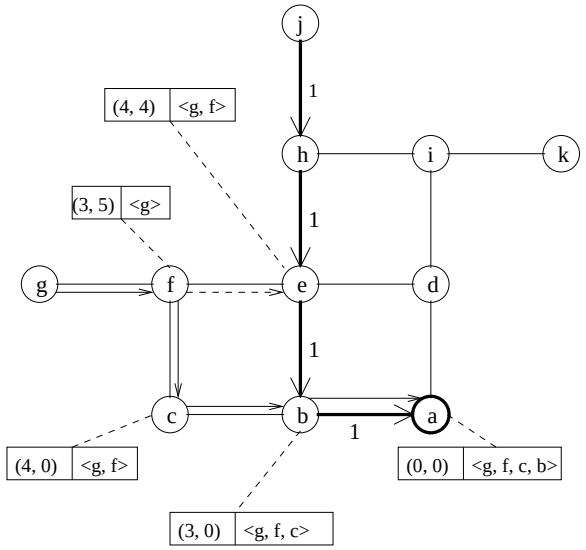


Fig. 3: Example of the tables maintained by the algorithm.

Example. Figure 3 illustrates a WMN where node a serves as a gateway. Function ICFAP_Find is assumed to be called with the following settings:

- The input vector $flow$ assigns a unit of flow to each of the four thick edges along the path from node j to the gateway a.
- The residual capacity vector cap is assumed to allow any of the flow augmentations mentioned below.
- The parameter s (the demand node) is set to g, and the rquired search is for an IC-FAP from g to the gateway a, with value $v_{req} = 1$ unit.
- The array $Nforward$ contains the following values:

$Nforward[g] = (f, 0)$, $Nforward[f] = (c, e)$, $Nforward[c] = (b, 0)$, and $Nforward[e] = (b, d)$.

Initially, all tables are empty. Subsequently, the search starts with node g, and considers a partial IC-FAP that sends one unit from g to f. Since the flow is assumed admissible, and $Nforward[f] = (c, e)$, the function computes the resulting loading factors at c, and e: $\ell(c) = 3$ (= $flow(e, b) + flow(b, a) + f(g, f)$), and $\ell(e) = 5$ (= $flow(j, h) + flow(h, e) + flow(e, b) + flow(b, a) + f(g, f)$), and inserts the entry $T_f[3, 5] = (g)$, where (g) is the initial part of the path (g, f).

Subsequently, the function considers extending the path from node f by sending a unit of flow to each of f's neighbours (other than g), leading to the following cases.

- Since the flow along the path (g, f, c) is assumed admissible, and $Nforward[c] = (b, 0)$, the function computes the resulting loading factor at b: $\ell(b) = 4$ (= $flow(h, e) + flow(e, b) + flow(b, a) + f(f, c)$), and inserts the entry $T_c[4, 0] = (g, f)$, where (g, f) is the initial part of the path (g, f, c).
- Since the flow along the path (g, f, e) is assumed admissible, and $Nforward[e] = (b, d)$, the function computes the resulting loading factor at b, and d: $\ell(b) = 4$ (= $flow(h, e) + flow(e, b) + flow(b, a) + f(f, e)$), and $\ell(d) = 4$ (= $flow(h, e) + flow(e, b) + flow(b, a) + f(f, e)$), and inserts the entry $T_e[4, 4] = (g, f)$, where (g, f) is the initial part of the path (g, f, e).

Next, the search continues from node c. Eventually, the search reaches the gateway a via b, and the path stored in $T_a[0, 0] = (g, f, c, b)$, with node a appended, is returned by the function. ■

We now describe the basic steps performed by the function in Fig. 4. Steps 1 initializes the IC-FAP table T_x at each node x to empty. Step 2 starts the search by setting x (the current node) to the input demand node s. The while-loop in step 3 iterates until an IC-FAP from s to t is found.

Step 3.1 has two nested for-loops: the outer loop expands the search by examining the neighbours of the current node x in a non-decreasing order of their distances to the gateway t. Note that the algorithm (in step 3.2) selects the first neighbour y of x for which an IC-FAP from s to y of value v_{req} is found for further extension. Hence, the above ordering gives preference to extending paths that terminate in close proximity of the gateway.

For a given neighbour y of the current node x, the inner for-loop in step 3.1 examines each of the potential partial IC-FAPs stored in T_x. The IC-FAPs are considered in a particular ordering of their associated keys. The ordering is defined by following relation: for two different keys (ℓ_1, ℓ_2), and (ℓ'_1, ℓ'_2), we write $(\ell_1, \ell_2) \leq (\ell'_1, \ell'_2)$ if $\max(\ell_1, \ell_2) \leq \max(\ell'_1, \ell'_2)$, or the maximum values are equal, and $\min(\ell_1, \ell_2) \leq \min(\ell'_1, \ell'_2)$. The paths are considered in a non-decreasing order of the above relation on the associated keys. So, a candidate partial IC-FAP from s to x receives higher priority if it minimizes the maximum loading factor at the forward nodes in $Nforward[x]$.

Function ICFAP_Find $(G, flow, cap, Nforward, v_{req}, s, t)$:

Inputs and Outputs: As described in Table I above.

1. For each node x in the WMN G, initialize table T_x to empty
2. Start the search from the current node $x = s$;
3. While (the gateway node t is not reached) {
   - 3.1 for each neighbour y of the current node x (in a non-decreasing order of the distances from the gateway), and each index (ℓ_1, ℓ_2) in the table T_x (in the order described in the main text) {
     - a. Let $P = T_x[\ell_1, \ell_2]$ be the stored candidate IC-FAP from s to x along which a flow increment of v_{req} units is possible;
     - b. If extending P by changing the flows on the directed edges (x, y) and/or (y, x) so as to obtain an IC-FAP from s to y is possible, then update table T_y accordingly; keep track of the first node y for which the above extension is possible;
   - 3.2 If a marked node y has been identified in the above step, then set the current node $x = y$, and expand the search for an IC-FAP further by continuing the while loop. Else (if no such node y is marked) then exit the while-loop;
   - }

   }
4. Return the IC-FAP stored at $T_t[0, 0]$;

Fig. 4: Pseudo-code for function ICFAP_Find

Step 3.1.b considers augmenting a path P stored in T_x with a link between the current node x, and one of its neighbours y. If the resulting augmented path satisfies the required flow constraints, and has the required value v_{req}, table T_y is updated with the new augmented path. Care is taken in implementing the above flow augmentation step so that any existing amount $flow(y, x)$ is first reduced as much as possible from the required value v_{req}, and then the possibly remaining amount is sent forward on the edge (x, y). Subsequently, if step 3.1 succeeds in identifying an IC-FAP from s to y, step 3.2 adopts the first such identified node as the current node x from which the search continues.

Running Time. To achieve efficiency in the running time, the function avoids performing exhaustive search, while maintaining awareness of the channel loading factors caused by sending new flows along the selected paths. In the worst case, the while-loop of step 3 iterates once for each node x in G. If we denote the maximum residual channel capacity at each of the $Nforward[x]$ nodes by $c_f(x)$, then the table T_x stores at most $c_f^2(x)$ paths. Additionally, if we denote by $d(x)$ the number of one-hop neighbours of x, then the for-loops in step 3.1 perform at most $d(x).c_f^2(x)$ iterations. In each iteration, step 3.1.b checks channel constraints at each node in G, thus each

iteration requires $O(n)$ time. The worst case total running time of the function is thus $O(\sum_{x \in V} c_f^2(x).d(x).n)$ time. Thus, if m is the number of links in G, and the maximum residual capacity at any node in the network is c_{max} then the running time is $O(n^2.c_{max}.m)$ time.

IV. Applications to Non-bifurcated Routing

Function ICFAP_Find described above provides a tool for tackling a variety of non-bifurcated routing problems on WMNs using conceptually simple algorithmic frameworks (e.g., greedy algorithms, search algorithms, etc.). In this section, we briefly discuss some of such approaches. The performance of the resulting algorithms, however, is a topic of current research.

We start by considering the gateway throughput maximization problem for single-channel wireless networks. Given the NP-completeness result of the single-channel problem, as mentioned earlier, the running time of any exact solution of the problem is expected to grow exponentially with the available channel capacity C, when flow demands assume arbitrary integer values in the range $[1, C]$. A simple framework for tackling the above maximization problem may compute the best result obtained by performing a number of iterations; each iteration starts by fixing an ordering of the set of all (node, flow demand) pairs: $\{(x, d) : x \in V, \text{ and } d \in D(x)\}$, and repeatedly calling function ICFAP_Find to search for an IC-FAP to route the flow demand d from x to the gateway g; each iteration terminates with a net gateway flow value obtained by serving as many flow requests as possible of the given ordering.

Likewise, for the more general problem where the network has a set GW of gateways, and each flow can be served by any gateway, a routing algorithm may start by fixing an ordering of the (node, flow demand, gateway) triplets: $\{(x, d, g) : x \in V, d \in D(x), \text{ and } g \in GW\}$, and repeatedly calling the function as mentioned above. The order of the triplets referred to above may be selected to satisfy some differentiated, or fair service criterion on the flows served from each node, and/or the total flow served by each gateway.

Tackling the more complex problem where the WMN has a number R of radio channels available at each node can also be approached in the above conceptually simple framework. Briefly, the method entails modifying function ICFAP_Find so that, for each of the available R channels, the input specifies the current flow vector, and the corresponding residual capacity vector. During each step in which the function tries to extend an IC-FAP by using a certain link (x, y), the function considers achieving this goal by using any of the available channels. That is, the joint assignment of flows to edges, and channels can be integrated within the IC-FAP search mechanism.

V. Performance Results

The non-bifurcated routing problem considered in the paper concerns WMNs where the transmission between contending flows is not synchronized in time. Our approach in tackling the problem adopts a set of linear constraints that work at the level of aggregate flows, where each flow is characterized by a requested data rate. The approach of computing routes in such contention environment based on dealing with traffic aggregates raises questions on the effectiveness of the obtained routes. Additionally, the requirement of avoiding traffic splitting is expected to contribute to lower achieved throughput, compared to utilizing routing schemes that do not impose such restrictions. In this section we explore the above performance aspects. In particular, we comment on the achieved average (over all flows), and maximum gateway throughput, and delay jitter, as described below.

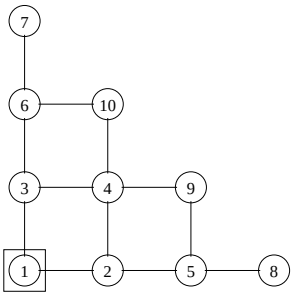


Fig. 5: The WMN topology

Simulation Environment and Parameters. Performance evaluation is done by implementing two complementary, but logically distinct, software modules: a flow-based algorithm implemented in C++ for route computations, and a network layer routing algorithm that works within the framework of the QualNet 3.9.5 [11] simulator. Our implementation of the non-bifurcated routing (NBR) algorithm employs a simple round robin selection scheme for choosing a flow demand to serve, the goal is to achieve fairness between nodes by maximizing the amount of served flow from each node. Performance of the NBR algorithm is compared with that of the standard Dynamic Source Routing (DSR) algorithm implemented in QualNet 3.9.5. We note that, in contrast to our algorithm, the operation of DSR does not impose the restrictions of non-bifurcated routing. Additionally, DSR is the basis of some recently proposed multi-channel routing algorithms (see, e.g., [6]).

Table II summarizes the important simulation parameters. The experiments use the network in Fig. 5, where node 1 serves as the gateway. In the network, routers are placed 100 meters apart from each other, and power control is used to set their transmission range to 129 meters. Subscriber units generate traffic flows. Each unit is placed 15 meters away from its serving backhaul router, and its transmission range is set to 29 meters. The experiments are done under the stringent conditions where all traffic flows (of end users, and the backhaul routers) contend for a single 802.11 channel. Each flow unit models a 40 Kbps of application layer traffic. Packets in each flow have size uniformly distributed in the range $[200, 300]$ bytes, and inter-arrival times uniformly distributed

in the range $[30, 50]$ milliseconds.

Data is gathered from multiple (typically eight) runs with each run having a simulation time of 10 minutes. The simulation time is long enough to extract stable performance from both routing algorithms.

TABLE II: Simulation Parameters

WMN Parameters	
Channel Capacity	100 units
Radio Range of Mesh Router	112 m
Radio Range of Subscriber Units	29 m
Maximum Subscriber per Router	10
Flow Demand per Subscriber	1 unit
Traffic Parameters	
1 Unit of flow	40 Kbps (Application Data)
Application-level Packet Size	Uniform: [200, 300] bytes
Packet Inter-arrival Time	Uniform: [30, 50] ms
Lower Layer Parameters	
Router Buffer Size for NBR	1000
Channel Bandwidth/Protocol	11 Mbps/ 802.11b
Data Acquisition Parameters	
Number of Runs per Data Point	8
Simulation Time of a Single Run	10 Min.

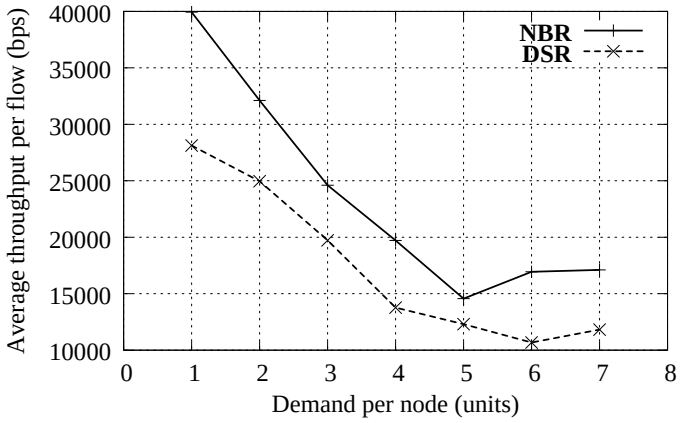


Fig. 6: Average throughput

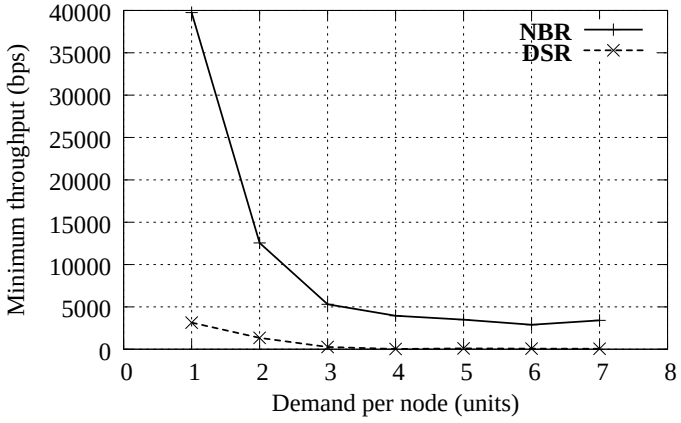


Fig. 7: Minimum throughput

Minimum and Average Throughput. We first measure the average (and aggregate) throughput perceived at the gateway under different system loads by means of varying demands of each subscriber units. Fig. 6 illustrates the achieved average throughput of all application layer flows received at the gateway. As can be seen, the NBR algorithm consistently improves on the DSR throughput by at least 20% at any input load.

The ability to allocate bandwidth to subscribers in a controlled way, so as to satisfy the service agreements, is a central and challenging issue in the design of WMNs. To assess the ability of the NBR algorithm to meet the above requirement, the algorithm has been equipped with a simple (but imperfect) mechanism to achieve fair service among all nodes in the network, as described above. Figures 7 shows throughput degradation of both algorithms as the offered traffic load increases. The potential benefit of the algorithm illustrated in the figure appears in ensuring non-zero traffic throughput for such discriminated against nodes in the network. In contrast, the DSR algorithm gives zero throughput.

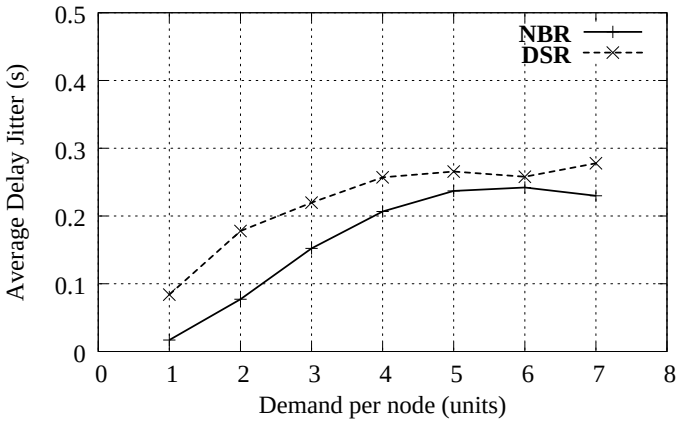


Fig. 8: Average Delay Jitter

Average Delay Jitter. Delay-sensitive traffic requires low delay jitter. A challenging task in the design of multi-hop wireless networks is to achieve relatively high throughputs, and simultaneously decreasing the average delay, and delay-jitter for a number of contending traffic streams. Figures 8, and 6 taken together illustrate that the NBR algorithm succeeds in improving over the DSR in both aspects simultaneously.

VI. Concluding Remarks

This paper considers the design of of non-bifurcated routing algorithms for serving traffic streams in multi-hop WMNs. In such network, each subscriber unit is assumed to have one, or more, streams to be routed to (or form) the network gateway(s) with no splitting at any hop. We formulate the routing problem as a network flow problem over wireless links that are subject to CSMA/CA contention, and devise a solution based on finding flow augmenting paths in the network. Simulation experiments using the QualNet 3.9.5 network simulator show that the devised routing algorithm produces competitive results for application layer traffic. In general, the results promote

further work on extending non-bifurcated routing to harness the capabilities introduced by newer wireless technologies.

REFERENCES

[1] IEEE 802.16-2004, "IEEE standard for local and metropolitan area networks - part 16: Air interface for fixed broadband wireless access systems," June 2004.

[2] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 73–87, ACM Press, 2005.

[3] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," in *IEEE Communications Magazine*, vol. 43, pp. S20–S30, IEEE, September 2005.

[4] M. Alicherry, R. Bhatia, and L. Li, "Joint channel assignment and routing for throughput optimization in multiradio wireless mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 1960 – 1971, November 2006.

[5] A. Raniwala and T.-c. Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, pp. 2223–2234, IEEE, March 2005.

[6] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 114–128, ACM Press, 2004.

[7] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom 2000)*, August 2000.

[8] P. Gupta and P. Kumar, "The capacity of wirless networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, March 2000.

[9] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, ch. 26, pp. 643–700. The MIT Press, 2nd ed., 2001.

[11] QualNet 3.9.5, Scalable Network Technologies. http://www.qualnet.com/.