Wireless Netw (2016) 22:2355–2367
DOI 10.1007/s11276-015-1102-1

Joint scheduling and routing using space–time graphs for TDM wireless mesh networks

Salik Warsi[1] · Vakul Jindal[1] · Saket Kumar[1] · Deepak Koli[1] · Amitabha Bagchi[1] · Vinay J. Ribeiro[1]

Published online: 29 October 2015

Abstract This paper presents a novel algorithm for joint routing and scheduling in TDM wireless mesh networks. We introduce a new construct, called a "space–time graph," which incorporates the spatial and temporal aspects of routing in one structure by replicating a spatial network connectivity graph in layers along the time dimension. The power of the space–time graph lies in the fact that a path from one node to another in it specifies both a physical route in space as well as a schedule in time for a message. Hence the complicated and intractable problem of routing and scheduling reduces to the relatively simpler problem of determining shortest paths in a graph. Through simulations we show that a simply greedy algorithm on the space–time graph outperforms two state-of-the-art methods in terms of time taken to successfully transmit a set of messages from their sources to their destinations.

✉ Vinay J. Ribeiro
vinay@iitd.ac.in

Salik Warsi
salikwarsi@gmail.com

Vakul Jindal
vakuljindal91@gmail.com

Saket Kumar
saket2201@gmail.com

Deepak Koli
deepakkoli93@gmail.com

Amitabha Bagchi
bagchi@cse.iitd.ernet.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India

Keywords Scheduling · Routing · Time-division multiplexing · Wireless · Mesh networks

1 Introduction

Wireless mesh networks (WMNs) are multi-hop wireless networks in which each node acts as a router to forward data between other nodes which may not be in range of each other. WMNs are particularly well-suited for applications requiring coverage of large areas, such as transportation systems, metropolitan area networks, surveillance systems, disaster management, battlefield scenarios etc., [2]. The importance of mesh technology is evident from the fact that various standards bodies including the 802.11 Wi-Fi and 802.16 WiMAX have developed standards for WMNs [18, 29].

Ensuring quality-of-service (QoS) in terms of delay and throughput is a challenging problem in WMNs. By their very nature, WMNs consist of hidden terminals, that is nodes which cannot hear each other. As a result, simple carrier sense techniques cannot prevent simultaneous transmissions from hidden terminals, thus causing packet collisions at other nodes. To prevent such collisions, packet transmissions must be scheduled across the entire WMN. Schedules can be decided by a central node with global knowledge of the network state, called *centralised scheduling*, or by the network nodes at large by exchanging messages between each other, a procedure called *distributed scheduling*.

In addition to scheduling, to ensure timely delivery of packets, one must find suitable routing paths for packets from source to destination within the WMN. Routing and scheduling are interdependent and together determine the QoS achieved. For example, a poor choice of routes can

lead to certain parts of the network getting overly congested, thus leading to large delays which scheduling cannot ameliorate. Intuitively, scheduling ensures that packet transmissions interfere less with each other in the time domain, and a good choice of routes ensures that packet transmissions interfere less with each other in the spatial domain. Both these hence together ensure that packets reach their destinations quickly.

In this paper we focus on WMNs which employ time-division multiplexing (TDM) at the medium access (MAC) layer. Here time is divided into non-overlapping time slots. A practical example of such WMNs are WiMAX mesh networks. We assume that all nodes are time-synchronised and are aware of when particular slots begin and end. Such synchronisation can in practice be achieved with the help of GPS clocks or through some time synchronisation protocol [13, 31]. We assume that one slot provides exactly the right amount of time to transmit a single packet from one node to any of its neighbours.

The problem we address is to minimize the total time taken to successfully send a set of given packets from their source nodes to their destination nodes using centralised scheduling. Different sub-problems of this problem have been shown to be NP-hard [14]. Variants of this problem have been studied in the literature [3, 4, 6–10, 15, 16, 19, 21, 30, 39, 40, 49]. One category of existing solutions solve only one of either scheduling or routing problems efficiently [8, 19, 34]. Another category of solutions formulate the joint scheduling and routing problem as an integer linear problem (ILP) or non-linear program which is intractable [3, 4, 6–8, 15, 30, 40, 49]. They then develop approximate solutions to the ILP in different ways (see Sect. 2 for details). Typically, these solutions cannot easily be modified to take as input online generated traffic, and instead require all the offered traffic to be known apriori. We compare our proposed algorithm to one representative example from each category of solutions mentioned above and show that it outperforms them. Moreover, our proposed solution naturally provides a solution for traffic generated online.

We approach the scheduling and routing problems jointly by defining a novel concept of, the *space–time graph*, that unifies space and time in a single structure and helps us approach scheduling and routing together in a natural way. To construct such a graph, we begin with a graphical model of the network in which each node represents one mesh node. We call this graph the connectivity graph of the network. To form the space–time graph of the network we stack connectivity graphs one on top of the other in a new dimension representing time. Each layer now represents the start of a time-slot. As we show in subsequent sections, a simple greedy algorithm that iteratively uses shortest path routing on the space–time graph solves both the routing and scheduling problems. The resulting solution is thus fast, elegant, and provides solutions comparable to, and in many cases outperforms, existing solutions to the problem we address.

We note that although we have used space–time graphs to solve a specific problem in WMNs, they are a general tool for modelling networks and we expect they will help solve various research problems in other areas such as software defined networks [43], delay tolerant networks [12, 33, 38, 48], mobile ad hoc networks (MANETs) [27, 46, 50], wireless sensor networks [25, 42, 44, 45], information centric networks [37], ATM networks [36], cognitive radio networks [20, 41, 47], and other heterogeneous networks [24, 51].

The rest of the paper is organised as follows. After describing related work in Sect. 2, in Sect. 3 we describe our network model and formulate the problem we solve. Section 4 presents our novel space–time graph algorithm. We present a few preliminaries used in validating our algorithm in Sect. 5 and two existing algorithms in Sect. 6. We compare our algorithm with others via simulations in Sect. 7 and conclude in Sect. 8.

2 Related work

The problem of developing a routing and scheduling algorithm for the transmission of messages in a graph where edges interfere with each other is well known. One of the earliest works by Tassiulas et al. [35] has argued that the problem must be decomposed into a pure routing and a pure scheduling problem for obtaining an optimal schedule and also that their results can be of importance in the topological design of a packet radio network.

Gabale et al. [14] gives a detailed survey of the various algorithms that attempt to solve our problem. Gore et al. [16] classifies the link scheduling algorithms into three classes and traces their merits and demerits. Most algorithms only solve this problem under a given set of constraints such as the graph being a tree or follow certain Quality of Service Constraints. In contrast, we solve this problem for a general graph. Also we realize that both routing and scheduling are important and intertwined aspects of this problem and present an algorithm that solves both the problems simultaneously, unlike several other methods with solve routing and scheduling one after another.

Several solutions solve only one of the routing or scheduling problems efficiently [10, 17, 19, 26, 28, 34]. Djukic et al. [10] gives a scheduling algorithm for this problem after routes have been specified but does not discuss how to obtain routes. Given an assignment of link bandwidths, the paper tries to develop an algorithm to find the minimum length TDMA schedule.

The iAWARE Algorithm dynamically allocates weights to edges to routing flows on less congested paths in the network [34]. Scheduling is not described in this paper. A similar approach is followed by Manikantan et al. [26].

Hajek et al. [17] have argued that the link scheduling algorithm can be solved in polynomial time. They focus mainly on the scheduling problem instead of joint routing and scheduling. They formulated the scheduling problem as a linear optimization problem and came up with an algorithm with running time $O(mn^4)$, where n is the number of nodes and m is the number of links.

Jain et al. [19] have proposed a linear programming approach to mainly try to solve the routing problem, assuming existence of an optimal scheduling algorithm. They present an idea of a conflict graph to formulate the equations of the linear program.

Another set of solutions formulate the problem as an Integer Linear Program (ILP) or non-linear program and develop approximate solutions in various ways [3, 8, 30, 40, 49]. Badia et al. [3] have tried to solve the problem of joint routing and scheduling in wireless mesh networks where the destination is a single gateway node. They model the problem as a integer linear program and then develop a genetic algorithm (GA) to solve the problem. They have shown that although the GA algorithm does not give optimal results, it is very close to optimal and it also scales well on large topologies. In contrast, we do not impose any constraint of gateway nodes.

Wang et al. [40] perform routing and scheduling in a mesh network consisting of ordinary wireless nodes and gateway nodes. Gateway nodes do not forward traffic between any other nodes, unlike ordinary nodes. All traffic is destined to the gateway nodes in the network. They follow a two-step process for routing and scheduling. First, routes are determined with the help of a linear program (LP) formulation and then messages are subsequently scheduled. More details are presented in Sect. 6.

Cruz et al. [8] have studied the problem of joint routing, link scheduling and power control in wireless networks. They use a primal dual approach for the problem. They present an algorithm that finds an optimal link scheduling and power control policy and then they construct a policy that allocates traffic rates on each of these links. They then use shortest path algorithms with the link weights set to certain "link sensitivities" to guide the search for a globally optimum routing.

Zhang et al. [49] have modelled the problem of joint routing and scheduling as a deterministic model under the contraints of interference among transmissions with the objective of reducing the overall system activation time. They approximate the problem using a linear programming problem and use a column generation approach to approximately solve it. In their numerical results they have shown that as the network becomes large and interference increases, gains can be obtained using multiple radios and multiple channels.

Molle et al. [30] formulate the joint scheduling and routing problem in wireless mesh networks consisting of gateway nodes as an integer linear problem. The problem is relaxed to obtain a linear program which is solved using a column generation approach. They have conjectured that an optimal route and schedule to gather traffic on gateways can be computed in polynomial time.

Several other papers address problems similar to the ones we address but with significant differences [4, 6, 7, 9, 15, 21, 39]. Kodialam et al. and Chen et al. [7, 21] tries to solve the same problem of joint routing and scheduling as we tackle, but while ignoring the secondary interference, that is interference caused by transmissions taking place in a close neighbourhood. Our work considers secondary interference.

Bhatia et al. [4] have studied the problem of joint routing. scheduling, and power control in a multi-hop wireless network with a single source and single destination node. We consider many sources transmitting to many destinations.

Capone et al. [6] have studied joint routing and scheduling when the nodes are equipped with directional antennas, which reduce interference, and a STDMA scheme is in place. Their models are based on mixed integer linear programs and are solved using a column generation approach. In their numerical results they have shown that directional antennas can greatly enhance the performance of WMNs.

Wang et al. [39] have used multi-packet reception (MPR) to reduce the negative effects of multiple access interference and therefore increase the capacity of an ad hoc network. They formulate the optimisation problem under a deterministic model and then given a heuristic aimed at approximating the optimal solution under the given constraints. Their numerical results demonstrate that the heuristic approach can exploit the MPR capability. Our work does not consider multi-packet reception.

Cui et al. [9] have studied the problem of joint routing and scheduling in networks where energy is a limited resource so that energy consumption needs to be minimised. They have proposed variable length TDMA schemes where they vary the slot lengths according to the requirements. They have shown that the problem can be approximated by converting it into a convex problem and solving it by already existing techniques. Their numerical examples have shown reduced energy usage and time delay. We consider only fixed slot lengths.

Girci et al. [15] have proposed link metric-based algorithms. They have designed a link cost metric which reflects the congestion on that link, power requirements and volume of traffic that has been delivered and then use a distributed Bellman–Ford algorithm to compute shortest paths. They have shown that this policy increases performance. However, in contrast to our work, they consider unlimited bandwidth in the network so that interference is not a problem.

Our idea of the space–time graph is inspired by the "geometrical view" of a packet routing problem on the line first presented by Adler et. al. [1] where the authors used a similar idea to analyse the approximation ratio of an algorithm for scheduling packets on a linear network.

3 Network model and problem formulation

In this section we describe our network model and formulate the problem we wish to solve.

3.1 Network model

We assume that each mesh router is equipped with a single radio which is permanently operating on a fixed channel. Our joint routing and scheduling algorithm can easily be extended to a multi radio, multi channel network, but for simplicity of the exposition we restrict ourselves now to a single radio, single channel assumption. We model a TDMA WMN as a static unweighted directed graph $G(V, E)$ consisting of a set of mesh nodes $V = \{v_1, v_2, v_3, \ldots, v_n\}$ and interconnecting edges $E = \{e_1, e_2, e_3, \ldots, e_l\}$ representing all directed communication links. We say that two communication links interfere if successful data transmission is not possible on both simultaneously due to mutual signal interference between them. All nodes are perfectly synchronised and time is divided into slots of equal size.

We use the notion of a *conflict graph* (see e.g. [19]), denoted by F_G, to model interference between communication links in G. Every edge in G is represented as a vertex in the conflict graph and an edge joins two vertices in F_G if and only if the corresponding edges in G interfere with each other. The conflict graph of the network of Fig. 1 is depicted in Fig. 2.

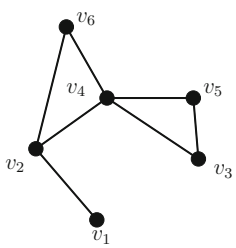

Fig. 1 An example network with 6 nodes

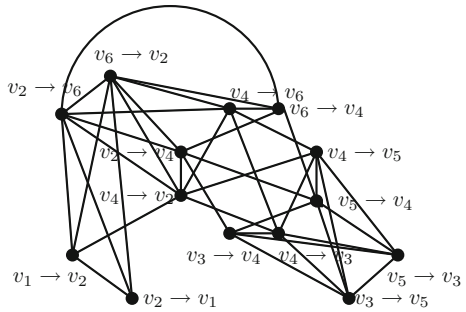

Fig. 2 The conflict graph for the network in Fig. 1

The above constructs are sufficient as inputs to our algorithm. Although in our simulation experiments we define G and F_G based on Euclidean distances, our algorithm does not have this as a requirement.

3.2 Problem formulation

We assume that at time $t = 0$ a set of messages $M = \{m_1, m_2, m_3, \ldots, m_K\}$ are to be transmitted from an arbitrary set of given source nodes to an arbitrary set of destination nodes. Each message can be transmitted in a single slot over an edge in G. Our goal is to determine routes for messages and schedule them so as to transfer all messages to their destinations in the minimum number of time slots.

4 Space–time graph algorithm

Using the network model G and its conflict graph F_G, we next present a novel concept of a *space–time graph* and then describe our joint routing and scheduling algorithm.

4.1 Space–time graph

In the space–time graph the node set of the physical network is replicated at every time slot (see Fig. 3). Hence if the physical network is represented by a graph $G = (V, E)$, the node set V_{ST} of the space–time graph S_G is $V_{ST} = V \times \mathbb{N} \cup \{0\}$ where time forms the second dimension.[1] For example, in Fig. 3 we have labeled the node depicting v_5 at time $t = 3$ as $v_{5,3}$. Note that the network in Fig. 1 has undirected edges, which captures the notion that transmissions are possible in either direction between nodes that are within transmission range of each other. But the space–time graph for this network (Fig. 3) has directed edges, which captures the idea that a path on this edge jointly

[1] In a practical implementation, the time-dimension is finite with maximum value T. If the initial value of T is found to be too small in the space–time algorithm, then it must be increased appropriately.

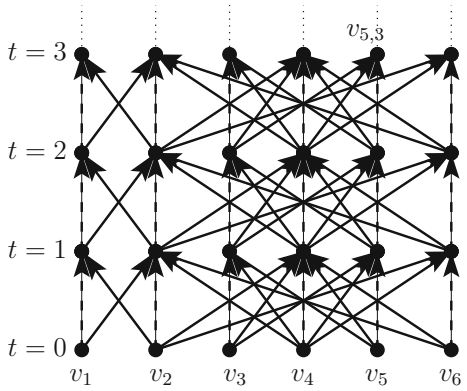

Fig. 3 The space–time graph of the network in Fig. 1

routes and schedules the packet and hence must always move forward in time.

The edge set of the space–time graph has two components. The first set of edges are those from $v_{i,t}$ to $v_{i,t+1}$ for all $v_i \in V$ and $t \geq 0$. These are depicted by dotted lines in Fig. 3 and represent the situation where no transmission takes place in the physical network i.e. sending a packet along this edge in the space–time graph corresponds to the node v_i keeping the packet in its buffer in the time slot between t and $t + 1$. The other kinds of edges, depicted by solid lines in Fig. 3 correspond to the case where a physical transmission takes place. These edges go from $v_{i,t}$ to $v_{j,t+1}$ where $j \neq i$ and (v_i, v_j) is an (undirected) edge in the physical network. For a packet to traverse one of these edges is equivalent to a packet being sent from v_i to v_j in the slot t to $t + 1$.

The power of the space–time graph lies in the fact that a "path" from one node to another in it specifies both a physical route in space, as well as a schedule in time for a message. Hence the complicated and intractable problem of routing and scheduling reduces to the relatively simpler problem of determining paths in a graph, the graph here being the space–time graph. To determine routing paths and a conflict-free schedule for a set of messages, we must hence find paths for the messages in the space–time graph from their respective source to destination nodes ensuring that transmissions do not cause collisions. To do so, our proposed solution, which we describe next, leverages well-known shortest path algorithms as well as the conflict-graph, F_G, of the network.

4.2 Space–time algorithm for joint routing and scheduling

Having constructed the space–time graph of the network, we proceed to describe the various steps of our joint routing and scheduling algorithm.

(Step 1) Ranking of messages All the messages $M = \{m_1, m_2, \ldots, m_K\}$ are ranked in decreasing order of priority with the highest priority messages being given a lower rank. The choice of ranking is entirely specified by the user of the algorithm. For example, in the case of online scheduling, one can rank the messages in the order in which they arrive at their source nodes. Another alternative is to rank messages randomly. Without loss of generality, assume that the subscript of a message represents its rank.

(Step 2) Space–time path for message Suppose we are given message m_p for any $p = 1, 2, \ldots, K$ which becomes available for transmission at its source node v_s in the space-graph at time t_p, and has destination node v_d. Now we set its source node to v_{s,t_p} in the space–time graph S_G. We create a virtual node v'_d to be its destination in the space–time graphs and create directed edges of weight 0 between $v_{d,t}$ and v'_d for all t.

Suppose we have found paths in the space–time graph (henceforth called *space–time paths*) for the first $p - 1$ (initially $p = 1$) messages and we need to find a space–time path for the pth message from its source to destination. We simply use a standard shortest-path algorithm such as Djikstra's Algorithm to find the shortest path, L_p, from v_{s,t_p} to v'_d. This ensures that m_p reaches its destination in the shortest time on the (pruned) space–time graph (see Step 3 for pruning), since the cost of a path in the space–time graph is equal to the number of time slots taken for the message to reach its destination. The resulting path gives us both a route for the message as well as the schedule for its transmissions over the air at various links in the network.

(Step 3) Pruning of space–time graph We must ensure that scheduling and routing of subsequent messages do not use any edges conflicting with the transmission of m_p at various links. To ensure this we prune S_G as follows. Recall that all the diagonal edges (that is from $v_{i,t}$ to $v_{j,t+1}$ where $i \neq j$) of L_p correspond to transmissions of m_p over the air whereas vertical edges ($i = j$) correspond to the case where the message m_p remains at the same physical node in a particular time-slot. We prune S_G by removing all diagonal edges of L_p from it. In addition, using the conflict graph F_G, for each of these diagonal edges we determine which other edges conflict with it *in that layer* in the space–time graph and remove those edges too. Note that an edge in one layer of S_G cannot conflict with an edge in any other layer as they correspond to transmissions in non-overlapping time-slots.

We illustrate this pruning with an example in Fig. 4. Suppose that in the network shown in Fig. 1 a packet is sent from v_1 to v_5 along the route $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5$ and that this packet is scheduled to leave v_1 at time 0 and reach v_5, without being delayed at any point, at time $t = 3$. The red

edges in Fig. 4 show the "path" this packet takes in the space–time graph. We note that in each level several edges that were present in Fig. 3 have now been removed. For example, in the first level the edge $(v_{2,0}, v_{4,1})$ has been removed, which is consistent with the fact (verifiable from the conflict graph shown in Fig. 2) that sending a packet from v_1 to v_2 causes interference with a packet being sent from v_2 to v_4 in the same slot. On the other hand an edge like $(v_{5,0}, v_{4,1})$ is retained since a transmission from v_5 to v_4 does not interference with a concurrent transmission from v_1 to v_2.

We increment p by 1 and repeat steps 2 and 3 until we find routes and schedules for all messages.

Remark 1 To break the ties between the various shortest paths possible in between the source and destination for each message in Step 2, for each diagonal edge of the space–time graph we use a tie-breaking weight equal to the number of its conflicting edges. Vertical edges have a tie breaking weight equal to 0 since they do not conflict with any other edge. Hence for each message we choose the shortest path which minimizes the sum of all tie-breaking weights among the shortest paths. This reduces the possibility of a later message being blocked as we are using or interfering with the smallest number of edges.

Remark 2 One advantage of the above algorithm is that it is similar to the random rank protocol (given in [32]) in that it has the property that if one packet ever blocks a second packet, then it always "stays ahead" of the second packet. This implies that second packet cannot ever block the first packet. In our algorithm, in each time slot, the packet with the highest priority is transmitted. Then we see if any of the other packets have a higher priority and schedule and route them. This prevents any circular interference where message m_i blocks m_j in one time-slot and then later gets blocked by m_j in another time-slot and so on. As shown in [32] this helps in optimizing the scheduling to a certain extent.

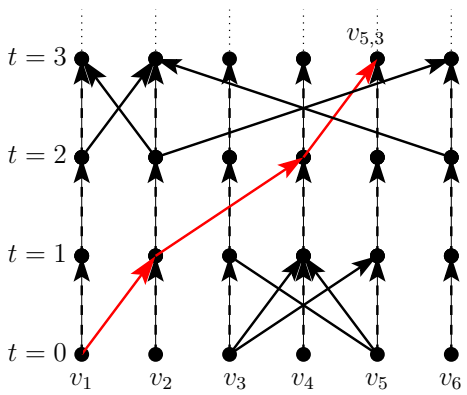

Fig. 4 The pruned space–time graph with a packet travelling $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5$

5 Validation preliminaries

In order to validate our space–time graph algorithm, we first compare it to simple solutions using a combination of standard routing and scheduling methods. We also present the notion of congestion and dilation which provide lower bounds on the performance of various algorithms.

5.1 Standard routing protocols

We employ the following two routing protocols in our simulations.

5.1.1 Shortest path routing (SP)

Given a directed weighted graph and a source-destination pair for a message, we find the shortest path between the source and the destination using Djikstra's algorithm. The weight of each edge is equal to 1.

5.1.2 Weighted shortest path routing (WSP)

Each edge of a given graph is assigned a weight of $1 + q\epsilon$ where q is the number of edges it interferes with and ϵ is a small positive tiebreaker ($\epsilon \approx 0$). The shortest path is then used to route each message given its source and destination node pair.

5.2 Standard scheduling protocols

Various scheduling algorithms we use in our simulations are as follows.

5.2.1 First come first serve scheduling (FCFS)

In a particular time slot, all messages are sorted according to how long they have been waiting at a particular node, with those having waited the longest being given the highest preference. Ties are broken randomly. We then forward messages in order of these preferences.

5.2.2 Furthest to go first (FTGF)

This method is similar to First Come First Serve Scheduling except that the messages are sorted with respect to the number of remaining hops which they need to travel. The message which has the most number of hops left is given the highest priority. After a message moves forward by 1 hop, the number of hops left for that message is decreased by 1.

5.2.3 Shortest to go first (STGF)

Similar to FTGF, the messages are sorted according to the number of hops they need to travel. The highest priority is given to message with least number of hops left to travel. The relative ranking of packets does not change in this protocol. To see why this is the case we apply an induction argument. If we assume that the ranking at time step k is the the same as that at time step 0, we note that any packet u that cannot move a step closer to its destination in step $k+1$ is delayed by a packet x with higher priority, i.e., by a packet x that is closer to its destination than u is. Since u was already further from its destination than x was from its own destination, u falls further behind and hence its priority remains lower. This argument can easily be formalised as a full proof.

5.2.4 Random rank scheduling (RR)

Each message is assigned a random rank which does not change over the course of the algorithm. When the network has to decide which message to send in a particular time slot, the ranks are used as priorities. We note that STGF simulates Random Rank Protocol to some extent as once given a rank in order of the path length, a message which has less number of hopes to cover, continues to have a smaller distance to cover due to it having a higher priority. The only difference is that here the ranks are not random but rather in order of their path lengths. This is different from FTGF where once a message gets transmitted over a hop, its priority can reduce. Hence in FTGF, the relative ranks can be changing while in Random Rank scheduling once the ranks are assigned, they are fixed.

5.3 Congestion and dilation

We use the notions of Congestion and Dilation to derive lower bounds on the time taken by any combination of routing and scheduling algorithms to solve the problem we address.

Given a wired connectivity graph and a set of paths on this graph, one for each message, we define the following terms [5, 32]:

- *Dilation* (D) It is the length of the longest path between a source and destination pair in a given graph.
- *Congestion* (C) Congestion for an edge is defined as the number of message paths which use the edge. The congestion of the graph is the maximum of the congestions for the different edges.

For a general graph and a corresponding path system returned by the routing algorithm, the scheduling algorithm has a natural lower bound of dilation. As a message can only be transmitted 1 hop in a time slot, the message with the longest path has to take at least D time steps. Also, each edge e will be used in C_e number of time slots, where C_e is the congestion for edge e. Therefore, the minimum number of time steps taken for completing all the transmissions is at least the maximum of Congestion and Dilation.

Now we perform a similar analysis for wireless paths taking into consideration interference between edges. Suppose the routing algorithm returns a set of paths P, one for each message. These set of paths form an overlay on our original connectivity graph. Hence given a connectivity graph and a path system we try to find the corresponding Wireless Congestion. The wireless congestion of an edge is defined as the sum of the number of message paths using it and the number of message paths using interfering edges. The wireless congestion of the network is the maximum of the edge wireless congestions. Note Dilation for wireless networks is the same as Dilation for wired networks.

With the help of an edge colouring problem on the conflict graph, we can obtain a lower bound on the Wireless Congestion. Two vertices in the conflict graph can be used in the same time slot only if they do not have an edge between them. Now we assign to each node e in the conflict graph a list of f_e different colours where f_e is the number of times edge e is used in P. All the colours assigned to a node (edge in the connectivity graph) must satisfy the constraint that they cannot be the same as a colour assigned to any of its neighbouring nodes. The smallest number of colours that can be assigned is a lower bound on the wireless congestion. The maximum clique is a lower bound on the smallest number of colours that can be assigned and is hence itself a lower bound on the wireless congestion.

6 State-of-the-art routing and scheduling algorithms for comparison

We compare our scheme with two state-of-the-art algorithms for routing and scheduling in WMNs described below.

6.1 iAWARE routing

The first method we compare our method with is iAWARE [34]. iAWARE efficiently finds routing paths for flows in a WMN keeping in mind the congestion experienced at various links in the recent past. By choosing routing paths for flows which circumvent congested spots in the network, iAWARE reduces queuing delays faced by messages, thereby improving overall throughput.

A node v_i of an edge e = (v_i,v_j) is assigned an Interference Ratio ($IR_e(i)$) which is equal to

$$IR_e(i) = \frac{SINR_e(i)}{SNR_e(i)} \quad (1)$$

where SINR is the Signal to Interference and Noise Ratio and SNR is the Signal to Noise Ratio given by

$$SNR_e(i) = \frac{P_i(j)}{N} \quad (2)$$

and

$$SINR_e(i) = \frac{P_i(j)}{N + \Sigma_{w \epsilon \eta(i)-j} \tau(w) P_i(w)}. \quad (3)$$

Here $P_i(j)$ is the signal strength from v_i to v_j, N is the background Noise, $\eta(i)$ is the neighbourhood of v_i, i.e. all vertices directly connected to v_i, $\tau(w)$ is the normalised rate at which node v_w generates traffic averaged over a period of time and is equal to 1 when node v_w sends out packets at the full data rate supported. As this period of time is not specified by [34] we let $\tau(w)$ be the normalised rate at which node v_w generates traffic since the start of the algorithm.

The Interference Ratio (IR_e) of an edge $e = (v_i, v_j)$ is the minimum of $IR_e(i)$ and $IR_e(j)$ as each packet transmission from v_i to v_j also involves the transmission of an acknowledgement from v_j to v_i. Since we ignore the acknowledgement in our simulations, we set $IR_e = IR_e(i)$. Finally we calculate the link metric iAWARE for an edge e as

$$iAWARE_e = \frac{ETT_e}{IR_e} \quad (4)$$

Here ETT is the expected number of transmissions and retransmissions required for a message to be sent across this edge. For a perfectly reliable edge as in our model this value is equal to 1. We set signal strength $P_i(j) = 1$ for all i and j and assume negligible noise N. This results in link metric $\Sigma_{w \epsilon \eta(i)-j} \tau(w)$. The weight of an edge is hence a measure of the number of messages sent by the neighbours of v_i.

6.2 LP based routing and scheduling

The second method we compare our algorithm with performs routing and scheduling in a two-step process [40]. First, routes are determined with the help of a linear program (LP) formulation and then messages are subsequently scheduled. The goal of the linear program is to maximize the throughput while keeping a minimum-fairness criteria for all the messages waiting for transmission. The algorithm models the network as a generic graph with source, destination and gateway nodes. Gateway nodes are the nodes connecting the network to the external Internet.

6.2.1 LP formulation

With maximising the throughput of the gateway nodes as the objective function, the algorithm uses the following constraints for formulating the linear program.

1. The difference of the outgoing flow and incoming flow at a node is equal to the amount of data transmitted by it into the network.
2. The flow transmitted by each node into the network should be greater than a minimum fairness amount proportional to the offered load of that node.
3. For each edge, the amount of flow entering and leaving are equal.
4. The *activity period fraction* of each edge with a non-zero flow should be in range [0, 1]. The activity period fraction of an edge is defined as the fraction of slots for which the edge has to be assigned in a particular given time period.
5. The sum of all activity period fractions for a set of interfering edges should be less than or equal to 1.

Each of the above conditions can be converted into a linear equation using appropriate variables and can be fed into an LP solver. After solving this linear program using a solver library, we get the flow through each node, the flow through each edge and the activity period fraction of each edge. The node flows can be used to identify the routes each message follows towards its destination. The value of the objective function determines how much data is to be transferred in one time period from the sources to the gateway.

6.2.2 Scheduling

Like in our paper, Wang et al. [40] assume that time is slotted. They try to determine a minimum slotting time period so that replicating the period multiple times minimizes the total transmission time of the messages. The scheduling is centralised and uses a greedy approach to schedule using the calculated routes. They schedule an edge in the earliest time slot available for it given that the start node of the edge has enough data packets to meet the flow requirement of the edge, as given by the linear program solution.

First, a conflict graph is created. The algorithm picks an edge with smallest degree and removes it and all its incident edges and pushes it to a list. This is repeated until the conflict graph becomes empty. Now the edges in the list are processed in the reverse order and scheduled one by one. A link is assigned the smallest time slot which has not been assigned to any of its neighbours in the conflict graph. The idea is to process those links earlier which have more number of interfering edges.

7 Simulations

We wrote our own C++ simulator to compare various scheduling and routing protocols. In our simulations, we first create a square plane of size 1 unit. Then we take n nodes and randomly place these nodes in the graph using an unbiased random number generator. Two nodes can communicate with each other if the distance between them is less than a given parameter δ called the transmission range. Two edges conflict if the transmitter node of either edge is within δ of the receiver node of the other.

We randomly choosing a source and destination for each of K messages. All messages are available for transmission at time zero.

We then compared the amount of time (number of time slots), it took to transmit these K messages in our space–time graph as well as in the various different routing and scheduling mechanisms.

For the following simulations the parameters were set as $n = 100$ and $\delta = 0.25$. For these parameters, the number of edges in the graph come out be around 1,500 which roughly depicts a dense graph. We first compare our algorithm to simple solutions using a combination of standard routing and scheduling methods described in Sects. 5.1 and 5.2. We then compare our scheme with the iAware and LP-based methods described in Sects. 6.1 and 6.2.

7.1 Shortest path routing

We first compared our algorithm with Shortest Path Routing coupled with various scheduling algorithms. The results in Fig. 5 clearly show that the space–time graph algorithm is able to outperform shortest path routing as the various messages are able to avoid congested nodes by taking separate and parallel routes to their destinations. Shortest Path Routing shows the best performance with Random Rank Scheduling.

7.2 Weighted shortest path routing

We next compared with Weighted Shortest Path Routing where the weight of each edge is equal to the number of edges that edge interferes with. We set $\epsilon = 0.01$. However, we observe in Fig. 6 that this leads to even worse performance than Shortest Path Routing. This is due to the fact that as the weights of the edges are static, all messages instead of taking their shortest paths prefer to take paths that pass through lighter (less weight) edges. Therefore congestion is instead caused at these lighter edges instead of the edges on the shorter path. Hence while dilation increases, congestion doesn't decrease much. Therefore this increases the amount of time taken to transmit the messages. Again, the weighted path routing has the best performance with the Random Rank Scheduling.

7.3 iAWARE routing

We next compare our algorithm with iAWARE [34]. As Subramanian et al. [34] does not mention any accompanying scheduling protocol, we implemented it along with the various different standard scheduling protocols. We were able to consistently perform better than iAWARE as seen in Fig. 7.

Since, iAWARE does not take into account scheduling and only takes care of routing, our method has an advantage over it. When we want to send many messages from a particular source to a destination iAWARE will make some of the later messages choose longer paths. However our space–time graph algorithm will recognise the fact that it is

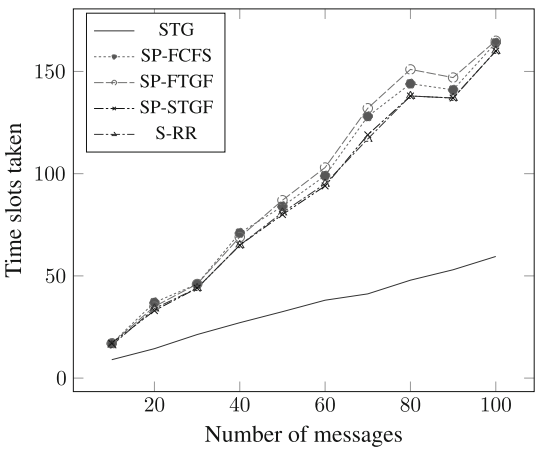

Fig. 5 Space–time graph versus shortest path routing. Space–time algorithm takes much less time slots to complete the same message transmissions

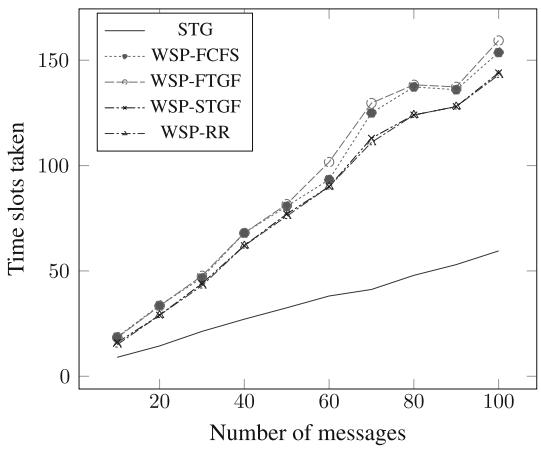

Fig. 6 Space–time graph versus weighted shortest path routing. Again, space–time algorithm takes less time to complete transmissions

beneficial for the algorithm to send all the messages on the same path one after the other. Using the following example, we illustrate why we perform better.

Consider the connectivity graph in Fig. 8 which consists of 2 vertices placed at endpoints and 2 paths connecting them. One path consists of $2n + 2$ vertices while the other consists of $n^2 - \epsilon$ vertices. Now let each odd vertex in the lower path send n messages to the next vertex in the path. Now the weight of the lower path will be equal to $n^2 + 2n + 2$. Also let the first endpoint vertex send 1 message to the final vertex. This message can be routed through either the upper path or the lower path. An online iAWARE algorithm having routed the first set of n^2 messages will now see a weight equal to $n^2 + 2n + 2$ for the lower path and $n^2 - \epsilon$ for the upper path and will hence send this message through the upper path. Now however this message takes $O(n^2)$ time to send the message. However our algorithm will first send the first set of messages in n time slots as they are all 1 hop message. The second message will now also be sent along the lower path in $2n + 1$ time slots. Hence in such a network we outperform iAWARE by an order. Note the dilation of the path system returned by iAWARE is itself $n^2 - \epsilon$, therefore it will take this much time irrespective of the scheduling algorithm used.

To further compare iAWARE with the space–time graph algorithm we conducted the following simulation on our random graph where we sent 100 messages from a group of sources located close to each other to a group of destinations also located close to each other but far away from the sources.

Hence we chose 2 nodes located at least a distance equal to 0.4 units apart. We now defined the Source Neighbourhood as the first node and its immediate neighbours and the Destination Neighbourhood as the second node and its immediate neighbours. We then selected 100 messages each having as its source a random node from the Source

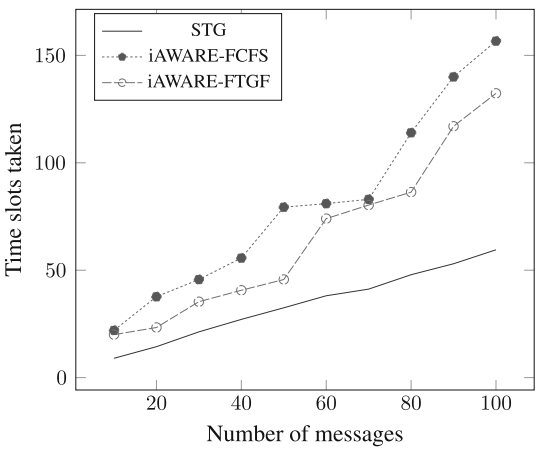

Fig. 7 Space–time graph versus iAWARE

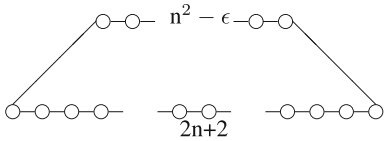

Fig. 8 Advantage of space–time graph algorithm (STG) over iAWARE

Neighbourhood and its destination as a random node from the Destination Neighbourhood. Now we transmitted these 100 messages first using iAWARE routing with Random Rank routing and then using the space–time graph algorithm. We observe from Fig. 9 that our algorithm outperforms iAWARE.

Here we can observe a clear advantage for our algorithm over iAWARE. In fact while we were unable to compute the exact Wireless Congestion of the paths returned by the iAWARE algorithm, we obtained a lower bound on it by computing the maximum clique of the conflict graph as explained in Sect. 5.3. In Fig. 9 we see that this lower bound is not much lower than the average number of time slots taken by our space–time graph algorithm. In fact in a few random simulation runs the maximum clique size found was more than the number of time slots taken by the space–time graph algorithm. Hence in these randomly generated cases, given the set of routing paths returned by iAWARE, no scheduling algorithm can perform better than the Wireless Congestion and will hence not be able to perform better than the space–time Graph Algorithm.

7.4 LP based routing and scheduling

Finally, we compare our space–time algorithm with the LP-based algorithm described in Sect. 6.2 [40]. The space–time algorithm clearly outperforms this algorithm as the number of messages increase (see Fig. 10). The LP-based algorithm implements routing and scheduling algorithms separately and assumes that the messages can be transferred in fractional amounts as given by the results obtained from solving the linear programming problem. To make the conditions same for both the algorithms, we assumed there is a single destination for all the messages instead of different source-destination pairs.

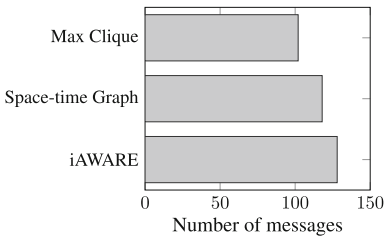

Fig. 9 Amount of time taken by STG and iAWARE compared with size of max clique in a network graph

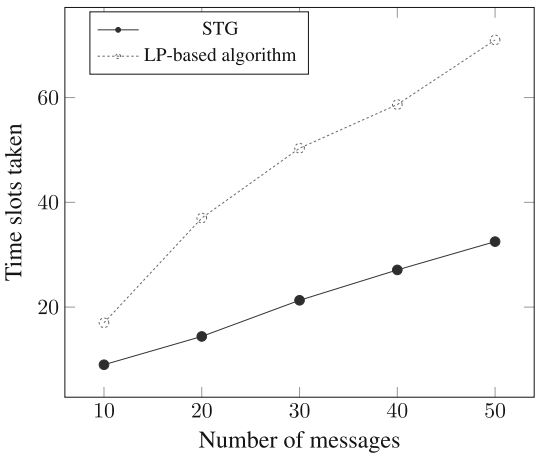

Fig. 10 Space–time graph versus LP-based scheduling and routing algorithm. The space–time algorithm performs better as number of messages increases

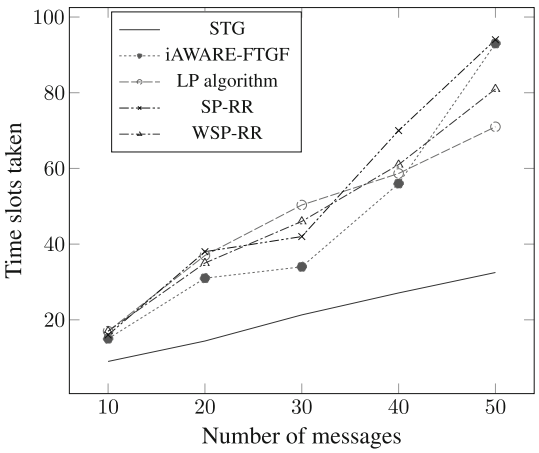

Fig. 11 Comparing best of all. The results clearly show outperformance of space–time algorithm

7.5 Comparing all of the above

Finally, we summarise our findings by comparing the best results of all the algorithms considered above in Fig. 11. Clearly our algorithm outperforms them, in some cases by almost 80 %.

8 Conclusions and future work

Using the novel construct of a space–time graph we developed a fast algorithm to perform both scheduling and routing in a TDMA wireless mesh network. Through simulations we showed that it outperformed two state-of-the-art competing methods.

Our present work can be extended in many ways. First, our algorithm can be modified for multi-channel and multi-radio networks [11]. This can be done by simply adding edges to the space–time graph for each new channel and changing the rules of interference appropriately. Second, one can study how the algorithm performs with finite buffers. If buffer sizes are finite $= B$, we can give the vertical edges of the space–time graph a maximum capacity of B instead of infinity, and remove them once the load on them reaches this capacity. Also these edges don't interfere with any other edge and hence the rest of the algorithm will remain the same. Third, one can extend the application of space–time graphs to multicast and develop network coding techniques on them [22, 23].

References

1. Adler, M., Rosenberg, A. L., Sitaraman, R. K., & Unger, W. (1998). Scheduling time-constrained communication in linear networks. In *Proceedings of the 10th annual symposium on parallel algorithms and architectures (SPAA'98)*. ACM, New York, pp. 269–278.
2. Akyildiz, I. F., Wang, X., & Wang, W. (2005). Wireless mesh networks: A survey. *Computer Networks*, *47*(4), 445–487.
3. Badia, L., Botta, A., & Lenzini, L. (2009). A genetic approach to joint routing and link scheduling for wireless mesh networks. *Ad Hoc Networks*, *7*(4), 654–664.
4. Bhatia, R., & Kodialam, M. (2004). On power efficient communication over multi-hop wireless networks: Joint routing, scheduling and power control. In *INFOCOM 2004. Twenty-third annual joint conference of the IEEE computer and communications societies*, Vol. 2. IEEE, pp. 1457–1466.
5. Busch, C., Kannan, R., & Vasilakos, A. V. (2012). Approximating congestion+ dilation in networks via Quality of Routing games. *IEEE Transactions on Computers*, *61*(9), 1270–1283.
6. Capone, A., Filippini, I., & Martignon, F. (2008). Joint routing and scheduling optimization in wireless mesh networks with directional antennas. In *IEEE international conference on communications, 2008. ICC'08*. IEEE, pp. 2951–2957.
7. Chen, L., Low, S., Chiang, M., & Doyle, J. (2006). Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *INFOCOM 2006. Proceedings of the 25th IEEE international conference on computer communications*, pp. 1–13.
8. Cruz, R. L., & Santhanam, A. V. (2003). Optimal routing, link scheduling and power control in multihop wireless networks. In *INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications. IEEE societies*, Vol. 1. IEEE, pp. 702–711.
9. Cui, S., Madan, R., Goldsmith, A., & Lall, S. (2005). Joint routing, MAC, and link layer optimization in sensor networks with energy constraints. In *2005 IEEE international conference on communications, 2005. ICC 2005*, Vol. 2. IEEE, pp. 725–729.
10. Djukic, P., & Valaee, S. (2009). Delay aware link scheduling for multi-hop TDMA wireless networks. *IEEE/ACM Transactions on Networking (TON)*, *17*(3), 870–883.
11. Duarte, P. B., Fadlullah, Z. M., Vasilakos, A. V., & Kato, N. (2012). On the partially overlapped channel assignment on wireless mesh network backbone: A game theoretic approach. *IEEE Journal on Selected Areas in Communications*, *30*(1), 119–127.
12. Dvir, A., & Vasilakos, A. V. (2011). Backpressure-based routing protocol for DTNs. *ACM SIGCOMM Computer Communication Review*, *41*(4), 405–406.

13. Elson, J., & Römer, K. (2003). Wireless sensor networks: A new regime for time synchronization. *ACM SIGCOMM Computer Communication Review*, *33*(1), 149–154.
14. Gabale, V., Raman, B., Dutta, P., & Kalyanraman, S. (2013). A classification framework for scheduling algorithms in wireless mesh networks. *IEEE Communications Surveys & Tutorials*, *15*(1), 199–222.
15. Girici, T., & Ephremides, A. (2002). Joint routing and scheduling metrics for ad hoc wireless networks. In *Conference record of the thirty-sixth asilomar conference on signals, systems and computers, 2002*, Vol. 2. IEEE, pp. 1155–1159.
16. Gore, A., & Karandikar, A. (2011). Link scheduling algorithms for wireless mesh networks. In *IEEE communications survey and tutorials*.
17. Hajek, B., & Sasaki, G. (1988). Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, *34*(5), 910–917.
18. Hiertz, G. R., Zang, Y., Max, S., Junge, T., Weiss, E., & Wolz, B. (2008). IEEE 802.11s: WLAN mesh standardization and high performance extensions. *IEEE Network*, *22*(3), 12–19.
19. Jain, K., Padhye, J., Padmanabhan, V., & Qiu, L. (2005). Impact on interference on multi-hop wireless network performance. *Wireless Networks*, *11*(4), 471–487.
20. Jiang, T., Wang, H., & Vasilakos, A. V. (2012). QoE-driven channel allocation schemes for multimedia transmission of priority-based secondary users over cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, *30*(7), 1215–1224.
21. Kodialam, M., & Nandagopal, T. (2003). Characterizing achievable rates in multi-hop wireless networks: The joint routing and scheduling problem. In *ACM Mobicom'03*.
22. Li, P., Guo, S., Yu, S., & Vasilakos, A.V. (2012). Codepipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, pp. 100–108.
23. Li, P., Guo, S., Yu, S., & Vasilakos, A. V. (2014). Reliable multicast with pipelined network coding using opportunistic feeding and routing. *IEEE Transactions on Parallel and Distributed Systems*, *25*(12), 3264–3273.
24. Liu, L., Song, Y., Zhang, H., Ma, H., & Vasilakos, A. V. (2015). Physarum optimization: A biology-inspired algorithm for the steiner tree problem in networks. *IEEE Transactions on Computers*, *64*(3), 819–832.
25. Liu, Y., Xiong, N., Zhao, Y., Vasilakos, A. V., Gao, J., & Jia, Y. (2010). Multi-layer clustering routing algorithm for wireless vehicular sensor networks. *IET Communications*, *4*(7), 810–816.
26. Manikantan, S. D., & Anjali, T. (2008). Load aware traffic engineering for mesh networks. *Computer Communications*, *31*(7), 1460–1469.
27. Marwaha, S., Srinivasan, D., Tham, C. K., & Vasilakos, A. (2004). Evolutionary fuzzy multi-objective routing for wireless mobile ad hoc networks. In: *Congress on evolutionary computation, 2004. CEC2004*, Vol. 2. IEEE, pp. 1964–1971.
28. Meng, T., Wu, F., Yang, Z., Chen, G., & Vasilakos, A. (2015). Spatial reusability-aware routing in multi-hop wireless networks. *IEEE Transactions on Computers*. doi:10.1109/TC.2015.2417543.
29. Mogre, P. S., Hollick, M., & Steinmetz, R. (2006). *The IEEE 802.16-2004 mesh mode explained*. Multimedia Communications Lab, Technische Universität Darmstadt, Technical Report KOM-TR-2006-08.
30. Molle, C., Peix, F., & Rivano, H. (2008). An optimization framework for the joint routing and scheduling in wireless mesh networks. In *IEEE 19th international symposium on personal, indoor and mobile radio communications, 2008. PIMRC 2008*. IEEE, pp. 1–5.
31. Noh, K. I., Serpedin, E., & Qaraqe, K. (2008). A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization. *IEEE Transactions on Wireless Communications*, *7*(9), 3318–3322.
32. Scheideler, C. (1998). *Universal routing strategies for interconnection networks* (Vol. 1390). Berlin: Springer.
33. Spyropoulos, T., Rais, R. N., Turletti, T., Obraczka, K., & Vasilakos, A. (2010). Routing for disruption tolerant networks: Taxonomy and design. *Wireless Networks*, *16*(8), 2349–2370.
34. Subramanian, A. P., Buddhikot, M. M., & Miller, S. (2006). Interference aware routing in multi-radio wireless mesh networks. In *2nd IEEE workshop on wireless mesh networks, 2006. WiMesh 2006*. IEEE, pp. 55–63.
35. Tassiulas, L., & Ephremides, A. (1992). Jointly optimal routing and scheduling in packet ratio networks. *IEEE Transactions on Information Theory*, *38*(1), 165–168.
36. Vasilakos, A., Saltouros, M. P., Atlassis, A., & Pedrycz, W. (2003). Optimizing QoS routing in hierarchical atm networks using computational intelligence techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *33*(3), 297–312.
37. Vasilakos, A. V., Li, Z., Simon, G., & You, W. (2015). Information centric network: Research challenges and opportunities. *Journal of Network and Computer Applications*, *52*, 1–10.
38. Vasilakos, A. V., Zhang, Y., & Spyropoulos, T. (2011). *Delay tolerant networks: Protocols and applications*. Boca Raton: CRC Press.
39. Wang, X., & Garcia-Luna-Aceves, J. (2009). Embracing interference in ad hoc networks using joint routing and scheduling with multiple packet reception. *Ad Hoc Networks*, *7*(2), 460–471.
40. Wang, Y., Wang, W., Li, X. Y., & Song, W. Z. (2008). Interference-aware joint routing and TDMA link scheduling for static wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, *19*(12), 1709–1726.
41. Woungang, I., Dhurandher, S. K., Anpalagan, A., & Vasilakos, A. V. (2013). *Routing in opportunistic networks*. New York: Springer.
42. Xiao, Y., Peng, M., Gibson, J., Xie, G. G., Du, D. Z., & Vasilakos, A. V. (2012). Tight performance bounds of multihop fair access for MAC protocols in wireless sensor networks and underwater sensor networks. *IEEE Transactions on Mobile Computing*, *11*(10), 1538–1554.
43. Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., & Vasilakos, A. V. (2014). Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, *20*(1), 4–18.
44. Yao, Y., Cao, Q., & Vasilakos, A. (2015). EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. *IEEE/ACM Transactions on Networking*, *23*(3), 810–823.
45. Yao, Y., Cao, Q., & Vasilakos, A.V. (2013). EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for wireless sensor networks. In *2013 IEEE 10th international conference on mobile ad-hoc and sensor systems (MASS)*. IEEE, pp. 182–190.
46. Yen, Y. S., Chao, H. C., Chang, R. S., & Vasilakos, A. (2011). Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs. *Mathematical and Computer Modelling*, *53*(11), 2238–2250.
47. Youssef, M., Ibrahim, M., Abdelatif, M., Chen, L., & Vasilakos, A. V. (2014). Routing metrics of cognitive radio networks: A survey. *IEEE Communications Surveys & Tutorials*, *16*(1), 92–109.
48. Zeng, Y., Xiang, K., Li, D., & Vasilakos, A. V. (2013). Directional routing and scheduling for green vehicular delay tolerant networks. *Wireless Networks*, *19*(2), 161–173.
49. Zhang, J., Wu, H., Zhang, Q., & Li, B. (2005). Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks. In *2nd International conference on broadband networks, 2005. BroadNets 2005*. IEEE, pp. 631–640.

50. Zhang, X. M., Zhang, Y., Yan, F., & Vasilakos, A. V. (2015). Interference-based topology control algorithm for delay-constrained mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, *14*(4), 742–754.
51. Zhou, L., Chao, H. C., & Vasilakos, A. V. (2011). Joint forensics-scheduling strategy for delay-sensitive multimedia applications over heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, *29*(7), 1358–1367.

Salik Warsi received his Bachelor of Technology in Computer Science and Engineering from IIT Delhi in 2013. He likes working with algorithms and is currently working as a software engineer.

Vakul Jindal did his schooling from B.G.S. Public School Barnala, Punjab and graduated with a B.Tech. in 2013 from IIT Delhi with Computer Science and Engineering as major. He is currently pursuing a career in the Indian Civil Services.

Saket Kumar received his Bachelor in Computer Science and Engineering from Indian Institute of Technology, New Delhi in 2014. His research speciality and interests include algorithms, computer networking and mathematics. He is currently working in the field of developing efficient algorithms and high-performance systems.

Deepak Koli is an undergraduate student at Indian Institute of Technology Delhi majoring in Computer Science. He is currently working on his major project in the field of software defined networking.

Amitabha Bagchi earned his Ph.D. in 2002 from Johns Hopkins University. His research interests range from network algorithms and data structures to the applications of random graph theory to wireless networks. Over the last few years he has been focusing on applying ideas from complex systems and stochastic processes to online social networks in a data-driven setting. Amitabha is an Associate Professor in the Computer Science and Engineering Department at IIT Delhi where he is a member of the Data Analytics and Intelligence Research group

Vinay J. Ribeiro is currently an Associate Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Delhi. He received his B.Tech. from I.I.T. Madras in Electrical Engineering and his M.S. and Ph.D. degrees from Rice University in Electrical and Computer Engineering. He received the best student paper award at the Internet Passive and Active Workshop 2003, a Texas Instruments Graduate Fellowship from Rice University, a Microsoft Outstanding Young Faculty Award from IIT Delhi, and is a member of Eta Kappa Nu. His research interests are in wireless networks, indoor positioning, and Internet of Things (IoT).