CORE: Centrally Optimized Routing Extensions for the IEEE 802.16 MeSH Mode

Parag S. Mogre, Nico d'Heureuse, Matthias Hollick, and Ralf Steinmetz
Multimedia Communications Lab, Technische Universitaet Darmstadt, Merckstr. 25, 64283 Darmstadt, Germany
Email: {parag.mogre,matthias.hollick}@kom.tu-darmstadt.de

***Abstract*—The IEEE 802.16 standard specifies a MeSH[1] mode which permits the deployment of Wireless Mesh Networks (WMNs) supporting carrier-grade QoS. The network operator for such planned WMNs is interested in maximizing the traffic admitted in the WMN and simultaneously supporting QoS. Recently network coding has emerged as a promising technique for increasing the throughput in WMNs. This paper proposes CORE, which addresses the problem of jointly optimizing the routing, scheduling, and bandwidth savings via network coding. Prior solutions are either not applicable in the 802.16 MeSH or computationally too costly to be of practical use in the WMN under realistic scenarios. CORE's heuristics, in contrast, are able to compute solutions for the above problem within a operator definable maximum computational cost, thereby enabling the computation and near real-time deployment of the computed solutions. We analyze the performance of CORE's heuristics via a thorough simulation study covering the typical usage scenarios for WMNs. The results presented demonstrate that CORE is able to increase the number of flows admitted considerably and with minimal computational costs. Further, the results provide insights into limiting factors for the gains which can be obtained in different usage scenarios.**

I. Introduction

WMNs are increasingly attractive for providing ubiquitous wireless network coverage from the network provider's point of view. The application scenarios for WMNs vary from extension of range for existing cellular networks to development of an independent network infrastructure for remote rural areas as well as formation of community mesh networks in urban regions (see [2] for a survey on WMNs). Recently, the attention in the research and standardization communities is turning towards supporting demanding multimedia traffic in the WMNs. The IEEE 802.16 standard's [10] MeSH mode and the ongoing IEEE 802.11s standardization incorporate sophisticated means to deploy WMNs supporting QoS for multimedia applications. In the current paper we focus on the IEEE 802.16 MeSH mode. The MeSH mode represents a paradigm shift in wireless medium access when compared to the contemporary IEEE 802.11 [9] standard. Sophisticated mechanisms are outlined in the MeSH mode to enable explicit reservation of bandwidth for transmission on individual links in the WMN permitting provision of QoS on a packet-by-packet basis. Provision of end-to-end QoS is not within the scope of the MeSH mode specifications. The network operator is interested not only in providing QoS but also increasing the amount of flows/traffic admitted in the WMN. Network coding [1] is a promising technique to achieve the latter goal in WMNs (e.g. [11]).

The optimal deployment of network coding in the MeSH mode requires the joint optimization of the routing, as well as the transmission schedule on individual links in the network, precisely reserving the amount of bandwidth required on each link. Prior work demonstrates the need for such a joint optimization (e.g. [15], [16]). However, the prior approaches found in the literature are not applicable to the MeSH mode or are computationally too expensive to be of practical use in realistic traffic scenarios (see Sec. II-B for a discussion of the related work). In the MeSH mode we want to route packets on routes satisfying the QoS requirements of the flows, and also route the packets on a single path to avoid jitter and reordering problems arising when multipath routing is used. Here, sufficient bandwidth (an integer number of minislots—the smallest unit of bandwidth allocation in the MeSH mode) needs to be reserved for the transmission (Sec. II-A provides an overview of the MeSH mode). The MeSH mode in contrast to the traditional IEEE 802.11 WMNs requires that the transmissions are scheduled in a contention free manner. The above constraints make the problem extremely hard, especially if we want to compute the optimal solutions and deploy the computed solutions in near-real time in a network with changing traffic demands. In this paper we present CORE (Centrally Optimized Routing Extensions) which is a framework intended to jointly optimize the routing, transmission schedule and bandwidth savings via network coding and be able to operate in dynamic WMNs. We presented a proof of concept for the CORE framework in [13]. There we demonstrated the ability of CORE's control messages to reconfigure the routing in the network in near real-time and its ability to adapt to changing traffic demands. In this paper we present the details of CORE's heuristics and thoroughly investigate the performance gains obtained using CORE and at the same time identify restricting factors for the gains obtained in typical usage scenarios for WMNs. To the best of our knowledge, this is the first work which looks at the problem of jointly optimizing the routing, scheduling and network coding in the IEEE 802.16 mesh.

In particular, our contributions are as follows:

- We present CORE (in Sec. III), which is able to work with standard routing protocols. CORE enables the adaptation of the routing tables at individual nodes in the WMN to achieve CORE's goal. Namely, jointly optimize the QoS aware routing, transmission schedule, and bandwidth

[1]We use the notation MeSH to refer to IEEE 802.16's mesh mode.

savings via network coding.

CORE uses heuristics (see Sec. IV) to approach its optimization goal. CORE is designed such that the network operator can parameterize CORE's heuristics to limit the computational costs to a given maximum threshold. This enables the computation of solutions for the optimization problem and the deployment of the computed solutions in near real-time, even in WMNs with dynamically changing traffic demands.

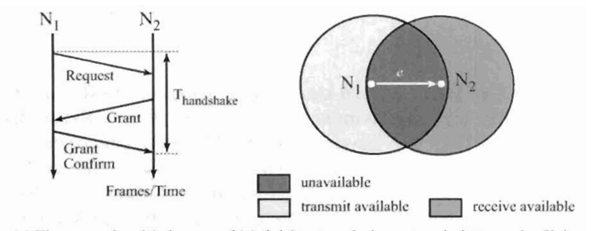
- CORE is designed such that distributed components (routing, distributed scheduling) are used to deploy the solutions optimized centrally. This implies that the WMN is able to continue with normal operation even when the central server running CORE's heuristics fails. This also means that the central optimization server used by core does not need to maintain complete global information (e.g. about the transmission schedule at individual nodes) and hence does not involve considerable overhead.
- We evaluate the quality of the solutions computed by CORE (see Sec. V) using a thorough simulation study covering typical usage scenarios for WMNs. The presented results demonstrate that CORE is able to achieve a considerable increase in the number of flows admitted in the WMN and with minimal computational costs. The results also highlight limiting factors for gains which can be obtained via jointly optimizing the routing, scheduling, and network coding in the most typical usage scenarios.

II. Background and Related Work

In this section we present a brief introduction to the MeSH mode. This is followed by an overview of relevant related literature.

A. MeSH Mode Background

The IEEE 802.16 MeSH mode [10] specifies the Medium Access Control (MAC) and the Physical (PHY) layers to enable the deployment of WMNs. The MeSH mode uses TDMA/TDD to arbitrate access to the wireless medium. The time axis is divided into frames. Each frame is composed of a control subframe and a data subframe. The data subframe is further divided into minislots (or simply slots). MAC layer messages meant for network setup and bandwidth reservation are mostly transmitted in the control subframe. Contention free access to the wireless medium in the control subframe can be both centrally regulated by a Mesh Base Station (MBS—a node usually providing access to external networks) or can be managed by the individual nodes (Subscriber Stations, (SS)) using the distributed mesh election algorithm specified by the standard (see [4], [10], [14]). Reservation of bandwidth for transmission of data messages in the data subframe can be both centrally managed by the MBS (called centralized scheduling) or a contention free transmission schedule can be negotiated by the nodes individually (termed distributed scheduling) without involving the MBS. Centralized scheduling is limited to scheduling transmissions on a scheduling tree specified by the MBS and rooted at the MBS. Distributed scheduling is more flexible and can be used to schedule transmissions on all the links (also those in the scheduling tree) in the WMN. Using distributed scheduling a SS negotiates its transmission schedule via a three-way handshake with the neighbouring node to receive the transmission (see Fig. 1). Given the limitations of centralized scheduling, we will, without loss of generality, assume that only distributed scheduling is used for the rest of this paper.



(a) Three-way handshake (b) Minislot state during transmission on edge/link e

Fig. 1. Distributed scheduling concept

Using the three-way handshake nodes can request and reserve a contiguous range of minislots for a contiguous range of frames (e.g. reservation $Resv(L_1, 1-10, 100-101)$ is used to denote that minislots numbered 1 to 10 are reserved for transmission on link with identifier L_1 for the frames numbered 100 and 101). The number of minislots reserved is termed as the demand level (denoted as $\Delta(MS)$) and the number of frames for which the reservation is valid is termed as demand persistence denoted here as $Per_{\Delta F}$, where ΔF is the number of frames for which the reservation is valid. Where as per the standard's specification $\Delta F \in \{1,2,4,8,32,128,\infty\}$. We may thus have reservations with demand levels 1... Max. Num. of Slots; and with demand persistences $Per_1, Per_2, Per_4, \ldots, Per_\infty$. Only slots reserved with persistence Per_∞ can be freed when no longer required via a cancel three-way handshake. For computing conflict free schedules, every node maintains the state for each minislot in each frame. Depending on the activities which may be additionally scheduled in a slot, the slot has one of the following states: *available* (*av*:transmission or reception of data may be scheduled), *transmit available* (*tav*:only transmission of data may be scheduled), *receive available* (*rav*:only reception of data may be scheduled), *unavailable* (*uav*:neither transmission or reception of data may be scheduled). Consider edge e=(N_1,N_2) $\in E$ in Fig. 1 (b), where E represents the set of edges in the WMN. Fig. 1 (b) shows how nodes in the network will update their slot states when a transmission is scheduled on edge e, provided all the nodes had the slots in state *av* at the beginning of the handshake. Neighbours of the receiver (N_2) overhear the grant and update the state for the granted slots to reflect that they may not transmit in the granted slots. Neighbours of the transmitter (N_1) overhear the grant confirm message and update their local slot states to reflect that they cannot receive any other transmission without interference in the confirmed slots. This process may be compared to the RTS/CTS mechanism used by 802.11

based nodes. A transmission may be scheduled on an edge e=(N_1,N_2) in a given slot *m* and frame *f* iff $s_m^f(N_1) \in \{av,tav\}$ and $s_m^f(N_2) \in \{av,rav\}$. Where $s_m^f(N)$ denotes the state of slot *m* in frame *f* at node *N*. We now define *I(e)* as the set of edges on which a transmission may not be scheduled (as per the states of the slots) considering that the slots which are reserved for transmission on edge *e* had status *av* prior to the three-way handshake for reserving the slots for edge *e*. We then define the blocking-cost of transmission on an edge *e* as $\varsigma(e) = |I(e)|$. Similarly, the blocking-cost for a path (route) r_{sd} between source s and destination d is defined as $\varsigma^{sd}(r_{sd})$ and is computed as shown in Eqn. (1).

$$\varsigma^{sd}(r_{sd}) = \sum_{e_i \in r_{sd}} |I(e_i)| \qquad (1)$$

Readers unfamiliar with the MeSH mode may find a detailed overview in Ref. [14].

B. Related Work

Our work is inspired by the seminal work of Katti et al. [11]. They demonstrate the substantial benefits which can be achieved by using a simple form of network coding [1] in WMNs. Ref. [16] builds on [11] by considering network coding and routing as a joint problem. The authors in [16] use a linear programming based approach to find an optimal solution in an 802.11 based MAC. However, the solution presented allows multipath routing and fractional bandwidth allocation on links. Further, the authors do not present any protocol to implement the presented solution in real WMNs. In particular, for reasons of avoiding jitter and out of order packet delivery issues we do not want to split packets belonging to a single flow along multiple paths. Additionally, it is not possible for us to reserve, e.g., 0.333 of the link capacity for a particular link in the WMN. Only an integer number of minislots may be reserved for transmissions on a link. The above constraints put our problem in the class of Integer Linear Programming (ILP) problems, which are generally considered to be NP-hard. We modeled our problem (see [7] for the details) as an ILP using the *GNU Linear Programming Toolkit* [8]. This open-source toolkit is able to solve ILP problems efficiently using a *branch-and-bound* approach. For small networks (7–10 nodes, 10–15 links, 2–5 flows) and very few (1-10) minislots, the ILP was solved within a few tens of seconds with reasonable results: correct routes where found and no collisions occurred. For larger networks (e.g. 16 nodes in a 4x4 grid layout, also more flows and/or links had similar effect), the solver was not able to find a solution in reasonable time (24 hours). Almost all of the work in the field of network coding assumes a 802.11 or similar MAC. No study is made for a sophisticated reservation based MAC like the MeSH mode. An important issue when using network coding is that it should not add a high delay penalty for the packets when coding is used (see [11] for arguments). Prior literature on wireless network coding makes coding decisions mainly on a packet by packet basis, which is not feasible without high delay in the MeSH mode. Reserving multicast bandwidth on a packet by packet basis via the three-way handshake in the MeSH mode is not only difficult (the receiving nodes have to agree to grant the same set of slots) but also involves the three-way handshake delay which can be considerable (see [4]), especially for multicast reservations. In the poster [13] we presented the concept for CORE and demonstrated the ability of its framework to operate in near real-time, jointly optimizing the routing and scheduling and network coding in the WMN. This paper builds up on [13] and studies the performance of CORE's heuristics in detail. Most of the other literature on scheduling and network coding is either restricted to multicast traffic (e.g. [15]), or does not permit spatial reuse for scheduling as specified for the MeSH mode (e.g. [5]). CORE is probably one of the first works jointly optimizing routing, scheduling and network coding in near real-time, and for realistic WMN sizes and traffic scenarios, esp. for the MeSH mode. Besides, the solution we present is deployed in a distributed fashion allowing the network to be resilient to failures of the centralized optimization entity.

III. CORE: Functional Overview

CORE has been designed to help the nodes in the WMN optimize the routes globally (and not from the individual node's point of view as done by contemporary shortest path algorithms). Here, CORE considers the QoS requirements of flows when adapting routes, and also strives to minimize the blocked bandwidth for a given traffic demand via use of network coding. CORE uses a central server to run its optimization heuristics (see Sec. IV) . Without loss of generality, for this paper we will assume that the central CORE server is also the MBS and hence will use MBS and CORE server with equal meaning. CORE's two fundamental design principles are: *Principle 1*: CORE should be able to work in realistic WMNs and be able to adapt the routes in response to dynamically changing conditions in the WMN in near real-time. Hence, CORE's heuristics are designed such that the network operator can limit the maximum computational effort spent on searching for an optimal solution. *Principle 2*: The usage of CORE's central server should be optional, i.e. nodes in the WMN should be able to deliver data to destinations even in the absence of the central server. Hence, CORE assumes the usage of distributed routing protocols and uses distributed scheduling to reserve bandwidth for transmissions on links in the WMN. We now explain CORE's working in a nutshell with the help of an example.

A. CORE Operation Explained

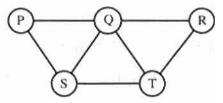


Fig. 2. Toy scenario to explain CORE's operation

Consider the toy-topology in Fig. 2 and two flows between the source destination pairs Flow 1:(S,R) and Flow 2:(R,P). Assume that initially only Flow 1 exists. Also assume that

Flow 1 uses route S–T–R. Now, when Flow 2 enters the network, the source (R) estimates the mean data rate for the flow and notifies the MBS (say node Q) of the new traffic demand while routing the data arriving from the new flow on the path computed by the distributed routing protocol. Assume that this route is R–Q–P. As the MeSH mode uses per-link encryption at the MAC layer no opportunistic listening and coding similar to that used in COPE [11] is possible. The MBS then uses the algorithms mentioned in Sec. IV to compute an optimal route combination for the flows. It may, for example, find out that routing the flows as Flow 1 along: S–T–R and Flow 2 along: R–T–S–P is a better solution which does not violate the QoS requirements and at the same time minimizes interference in the network and also generates opportunities for deploying network coding (e.g. at node T) in the network further reducing needed transmissions. The MBS now sends control messages to the affected nodes notifying them of the required routing table updates and the schedule changes. In this example, the MBS informs nodes R, T, S, about the new routes computed, the nodes then change their individual routing tables accordingly. The MBS also notifies node T about the possibility for deploying network coding. However, just changing the routing tables is not sufficient in the MeSH mode as the IEEE 802.16 MeSH mode uses explicit bandwidth reservation for data transmission. As there may not be sufficient bandwidth already reserved on the new route, new packets arriving will have to wait in the queue until enough bandwidth has been allocated on the new route. At the same time, the bandwidth on the old route might be unused. Consequently we also need an interface from the routing layer to the MAC layer, to allow the routing to directly initiate the request or cancellation of bandwidth. The MBS using the CORE mechanism thus notifies the individual SSs of both the routing changes as well as changes needed to the reservations on the links affected by the routing changes. To ensure a smooth transition to the new constellation the MBS specifies a frame number after which these changes take effect.

To further reduce overhead of CORE's centralized control and centralized algorithms, we distinguish between long-term (with persistence Per_∞) and short-term reservations (reservations with persistences $< Per_\infty$). The MBS is periodically (or when a new flow arrives, or the mean demand level changes drastically) notified of the mean demand level of the traffic demands by the sources using the periodic network configuration messages transmitted in the control-subframe. The computations at the MBS use this as the traffic demand. The MBS then computes the routes and instructs the individual SSs to reserve bandwidth (an amount corresponding to the demand on the link) on the concerned links using Per_∞ (good until cancelled or reduced) reservations. The traffic demand in a real network is by no means constant at the mean level and may show fluctuations and bursts of data. The SSs, when using CORE, reserve bandwidth for such traffic bursts using short-term reservations without needing to contact the MBS. This enables a quick response to bursts of traffic with low overhead. To ensure that minislots are available for such short-term reservation CORE specifies a maximum fraction of the data-subframe (i.e. number of minislots) which may be used by the centralized optimization mechanism of CORE for long-term reservation. The remaining minislots are available for short-term reservation. Thus, the centralized optimization part of CORE, when computing the maximal schedule and its feasibility considers that it has only the number of minislots per data-subframe as are permitted by the network operator.

From the overview we see that CORE optimizes the network centrally, however, uses distributed means to deploy the optimized solution. Additionally, it has been designed such that failure of CORE's centralized server does not lead to complete breakdown of the WMN. Ref. [7] provides details about the control protocols designed for CORE and the extensions we made to the IEEE 802.16 MeSH mode.

IV. CORE: Details of the Heuristics

To make the optimization problem tractable, and to enable the design of heuristics for finding a solution in real-time, we split the problem into several subproblems. These are: Route Preselection, Optimal Route Combination (*OptRC*) and Maximal Scheduling (*MaxSch*). We now describe the above subproblems and our solutions to each.

A. Route Preselection

The goal of this heuristic is to limit the number of routes per flow that are considered by the OptRC heuristic to an operator-defined value (κ). This routine gets all the routes for a flow which are feasible w.r.t. the QoS constraints. From this set of routes per flow, the best (w.r.t. the route blocking-cost as specified in Eqn. (1)) κ routes are retained. This gives a set of at most κ routes per flow which are feasible w.r.t. QoS and, at the same time, potentially block a minimum number of links due to data transmissions on the path. The selected routes are then the input for the OptRC heuristic.

B. Heuristic for the OptRC Subproblem

The solution to the OptRC problem uses the solution to the MaxSch problem as a subroutine (within the procedure *combine*() shown in the pseudocode for Algorithm 1). The OptRC problem can be formulated as follows. "*Given a set of flows, their traffic demands, and a set of routes per flow, what is the optimal combination of routes (by choosing one route per flow) such that: the maximum traffic demand can be supported in the network, the overall interference in the network is minimized when using the schedule produced by the MaxSch subroutine, and bandwidth savings via network coding are maximized?*"

A subset of the flows from the given set of flows may be dropped if they cannot be scheduled using the MaxSch routine for lack of sufficient capacity (minislots). In particular, the solution to the OptRC problem is a set of routes which contains a single route for each source-destination pair. Algorithm 1 shows our proposed algorithm as pseudocode. Instead of finding an optimum route set for all flows at once, the algorithm operates stepwise. We next define how Algorithm

Algorithm 1 OptRC Algorithm

```
Definitions:
  $f$: List of flows to be processed, ordered according to the flow's importance.
  $f_0$: most important, unprocessed flow (i.e. first element of $f$).
  $a$: Set of flows for which a route has already been found.
  $p$: A partition, i.e. a self sorting list of flows; ordered according to the flow's importance.
  $P$: Self sorting list of partitions (i.e. list of lists of flows) with elements $P_i$; ordered by the importance of the first flow of the elements.
  $P_0$: First element of $P$, i.e. the partition containing the most important, unprocessed flow.

 1: $a \leftarrow \emptyset$
 2: repeat
 3:     $p \leftarrow \emptyset$
 4:     $P \leftarrow \emptyset$
 5:     $comb \leftarrow 1$
 6:     while $comb \cdot k_0 \leq \Delta$ do                 ▷ Add flows to $p$ until $comb > \Delta$
 7:         $comb \leftarrow comb \cdot k_0$
 8:         $p \leftarrow p \cup \{f_0\}$
 9:         $f \leftarrow f \setminus \{f_0\}$
10:     end while
11:
12:     $P \leftarrow P \cup p$
13:     repeat                                                ▷ Binary search
14:         $best \leftarrow$ combine($P_0 \cup a$)             ▷ Search best set of routes
15:         if $best = \emptyset$ then                       ▷ No valid combination found
16:             if $|P_0| = 1$ then                          ▷ $P_0$ contains only one element
17:                 $P \leftarrow P \setminus \{P_0\}$              ▷ Flow in $P_0$ not routeable
18:             else                              ▷ Split current partition in two partitions
19:                 $np \leftarrow \left\{P_{0,x} : x \geq \left\lfloor \frac{|P_0|}{2} \right\rfloor\right\}$
20:                 $P_0 \leftarrow P_0 \setminus \{t : t \in np\}$
21:                 $P \leftarrow P \cup \{np\}$                ▷ $np$ has second position in $P$
22:             end if
23:         else
24:             $a \leftarrow a \cup P_0$
25:             $P \leftarrow P \setminus \{P_0\}$
26:         end if
27:     until $P = \emptyset$
28:     fixRoutes($best$)                 ▷ Fix the flows to the routes currently found
29: until $f = \emptyset$
```

1 divides the OptRC problem into smaller optimization steps. From the list of yet to be routed (non-processed) flows f, t flows are chosen, such that

$$\prod_{s=1} k_s \leq \Delta \tag{2}$$

where Δ is a constant set by the network operator and k_i is the number of possible routes of the ith non-processed flow (Lines 3–11). These t flows form one partition p as shown in Algorithm 1. In our algorithm we add flows to a partition till the bound given by Eqn. (2) is reached. We always select the first flow (f_0) in the list of flows f as the next flow to be added to a partition of flows to be optimized in parallel. Hence, the order of flows in f plays a crucial role in the performance of Algorithm 1.

For the results in this paper, we sorted the flows based on their traffic demand and they are listed in f such that a flow which has a higher traffic demand occurs before flows with lower traffic demands. Optimizing a larger number of flows in parallel is beneficial, because it enables the computation of a schedule, which allows maximal concurrent transmissions for the considered flows, by adapting the routes for all these flows. Thus, the main goal of Algorithm 1 is to efficiently search for a combination of routes which allows the maximum traffic demand to be supported. Algorithm 1 uses binary search within a given partition to quickly find the subpartition (P_0) of flows which can be jointly scheduled when the given partition of t flows cannot be jointly scheduled (see lines 13–27).

The procedure *combine*() passes all possible route combinations for the given flows as argument to the MaxSch algorithm and returns the best schedulable set of routes. The binary search enables us at the same time to find out flows which cannot be scheduled by the MaxSch routine together with the flows which have been already scheduled and for which the routes have already been fixed. Flows for which no schedulable solution could be found are excluded from any further calculations. All other flows regarded in this step are fixed to the route that has just been found. This procedure repeats until all flows have been considered. In the case that each of the t flows has κ routing possibilities, the maximum number of route combinations $\hat{C}(\kappa, t)$ that have to be checked (implemented using the *combine*() procedure which calls the computational expensive MaxSch algorithm for each possible route combination) until all t flows have been processed can be calculated iteratively by

$$\begin{aligned}
\hat{C}(\kappa, 0) &= 0 \\
\hat{C}(\kappa, 1) &= \kappa \\
\hat{C}(\kappa, 2) &= \kappa^2 + 2\kappa \\
&\dots \\
\hat{C}(\kappa, t) &= \kappa^t + \hat{C}(\kappa, \lceil t/2 \rceil) + \hat{C}(\kappa, \lfloor t/2 \rfloor).
\end{aligned} \tag{3}$$

One can see from Eqn. (3) that the number of possible route combinations and, thus, the number of schedules that have to be calculated, increases exponentially. The operator can choose between processing either more flows in parallel or allowing more routing alternatives for each flow.

C. Heuristic for the MaxSch Subproblem

The MaxSch heuristic is given a set of flows and one route per flow. It first computes the bandwidth required (demand) per link in the network for the given set of flows and routes. The MaxSch next attempts to find a maximal schedule for the demand, returning the amount of (minislot,link) tuples blocked by the computed schedule. The returned value can, in general, be any metric which allows the calling function to evaluate the quality of the computed schedule. In case the given set of flows cannot be scheduled due to bandwidth limitations, an error value is returned. The MaxSch subroutine is called repeatedly by the *combine*() procedure shown in Algorithm 1, to find the best (minimum blocking) route combination.

For our subproblem, a maximal schedule is a set of scheduled data transmissions (for the given traffic demand) per minislot such that no further non-conflicting data transmissions can be scheduled. The above definition is similar to the definition for the maximal slot assignment presented in [6]. To find a maximal schedule we use an adapted version of the greedy maximal scheduling algorithm presented in [12] (similar scheduling algorithms may also be found in [6], [17]). The MaxSch heuristic assigns the first minislot to the link with the highest demand. Thereby, a certain set of links cannot be activated in the same minislot. Of the remaining links, again the link with highest demand is chosen and assigned to the current minislot. The procedure repeats until no more links can be activated in this minislot. The demand of all active links is then reduced by one and the heuristic restarts with the updated demands per link for the next minislot.

We slightly adapted the MaxSch heuristic described above so that it can also be used for scheduling network coding transmissions (which are multicast transmissions in contrast to the normal unicast transmissions in a WMN). In the presence of per-link encryption in the WMN (as assumed), network coding via opportunistic listening as outlined in [11] is not possible. Hence, network coding in our scenario is only possible when we have a traffic setup similar to the "Alice-Relay-Bob" scenario outlined in Ref. [11]. We use the flow and routing information to determine all possible network coding opportunities and create a virtual network coding link for each of them. The demand on these virtual network coding links is equal to the minimum of the demands of the corresponding two unicast transmissions, which are now replaced by transmissions on the virtual network coding link. Since some of the demand is now served by the network coding links, the demand on the corresponding unicast is reduced by the same amount. However, as a single network coding transmission can transport the double amount of data compared to a normal transmission, network coding links should be preferred by the MaxSch heuristic. Consequently, when choosing a link to schedule next, we consider the effective demand for network coding links to be twice the real demand (in minislots).

V. Evaluation

We next evaluate CORE's heuristics via monte-carlo simulations (CORE's functionality was implemented into an extended version of the JiST/SWANs simulator [3]). Due to space limitations, we restrict the discussion to the study of the network performance for the mesh topology shown in Fig. 3 for two distinct simulation setups. Additional experiments (different topologies, different sorting for the flows for *OptRC*) can be found in [7] and show similar performance gains using CORE. In Setup I we compare the quality of the solution obtained using CORE vs. the optimal solution (considering all possible route combinations in the working set of the heuristic). In Setup II we analyze the performance of CORE for different usage scenarios of the WMN. In particular, we investigate different traffic distributions to model an access network, an enterprise/community network and a mixture of both. As a baseline for our joint routing and scheduling heuristic, we use standard shortest-path routing using either hop-count or the blocking-cost of a path as defined in Eqn. (1) as routing metric.

A. Simulation Results: Setup I

To find the globally optimal solution we implemented a brute-force algorithm which explores all feasible route combinations. Due to the prohibitive computational costs involved for the brute-force search we limited the study in Setup I to only 6 flows in the network with a limit of 20 minislots

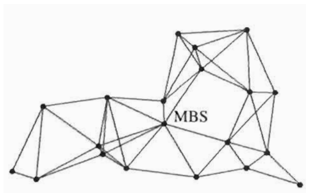


Fig. 3. Simulated 20 node WMN topology with Mesh Base Station (MBS)

in the data-subframe. The traffic demands for the individual flows were uniformly distributed between 2–7 minislots per frame per flow. The source destination pairs were randomly selected such that trivial flows were not generated (i.e. the minimum hop-path is of at least two hops). We performed 400 replications of the experiment. The following algorithms have been studied:

- Minimum-hop routing (`MiH`)
- Minimum-blocking path routing (`MiI`, see Eqn. (1))
- CORE's routing heuristic (`He`)
- Optimal routing using brute-force (`BF`)

We analyzed setups with and without network coding (`NC`/`No NC`). In the `NC` case, network coding opportunities are recognized by the scheduler and the corresponding slots for the (multicast) transmission are reserved.

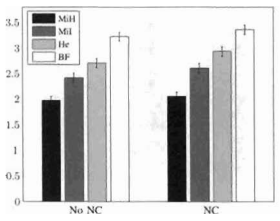
Fig. 4(a) shows the average *number of flows which could be scheduled* (we show the 95% confidence intervals if not noted otherwise). Using the optimal `BF` algorithm on an average less than 4 of the 6 flows offered could be scheduled, i.e. we operate the network in saturation. Fig. 4(a) shows that the shortest-path routing `MiH` performs worst. `MiI`, the second best scheme, outperforms `MiH` because the selection of minimum-blocking paths frees network resources, thus enabling the scheduling of additional flows. This result acknowledges the importance of choosing interference-aware routes in the WMN. The developed heuristic `He` performs even better. Infact, the increase in performance compared to `MiI` indicates the performance gain that can be realized by optimizing the routes for all flows in parallel. Moreover, the performance of `He` is close to 90% of the optimal performance of `BF`, if we consider the number of admitted flows.

The obtained results with network coding are inline with our earlier findings for the relative performance differences of the analyzed schemes. In absolute terms, despite the fact that we only offer 6 flows, the creation of network coding opportunities helps reduce the number of transmissions, thereby leading to fewer blocked links per minislot and, thus supporting more flows. We see that the performance of `He` is even closer to the optimum compared to the non-network coding case.

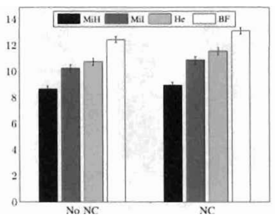
Fig. 4(b) shows the average *served total traffic demand* (in minislots required per frame) for the scheduled flows. It can be seen that `He` supports a larger traffic demand compared to `MiH` and `MiI`. Again, `BF` gives the optimum performance that can be achieved. Fig. 4(c) shows the number of (link, minislot) tuples *blocked per scheduled demand*, which describes the bandwidth efficiency of the scheme. A lower value for this metric indicates a higher efficiency of the scheme.

TABLE I
SEARCH EFFORT, SUCCESSFUL REROUTES AND ESTABLISHED NETWORK CODING SESSIONS FOR SETUP I

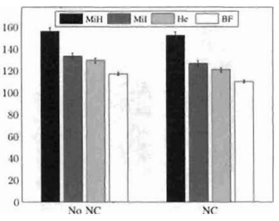
	Combinations Tried		Reroutes		NC Sessions	
	No NC	NC	No NC	NC	No NC	NC
MiH	6 ± 0	6 ± 0	0	0	0	0.38 ± 0.08
MiI	6 ± 0	6 ± 0	0	0	0	0.95 ± 0.14
He	74 ± 1	72 ± 1	0.845 ± 0.08	0.95 ± 0.09	0	1.49 ± 0.15
BF	15624 ± 0	15624 ± 0	1.15 ± 0.10	1.24 ± 0.10	0	1.83 ± 0.15



(a) Number of flows scheduled



(b) Demand served



(c) Blocked per demand

Fig. 4. Simulation results for Setup I

This translates into a higher probability that available (link, minislot) tuples still exist in the network for a fixed demand; these available resource can in turn be used for scheduling additional data. Only BF is able to outperform He.

Table I shows the number of route *combinations tried* (searched) by the individual algorithms to achieve the results shown in Fig. 4. The value *reroutes* represents the number of times a flow was routed on the non-default path (which is assumed to be the MiI path). Given, the small number of flows in Setup I we do not have a large number of reroutes. It can be seen that the He and BF algorithms reroute more flows in order to establish network coding opportunities, thereby conserving bandwidth in the network. The value *NC sessions* counts the mean number of network coding sessions established (a network coding session is defined as a tuple (a, X, b) where a, X, and b are nodes and X acts as a network coding relay for packets from a to b and vice versa). Even for the small setup, our algorithm He is able to establish significantly more NC sessions than the baseline algorithms.

B. Simulation Results: Setup II

In Setup II we analyze the performance of our heuristic for different traffic patterns, representing the following typical usage scenarios for WMNs:

- Operation of the WMN as a wireless access network is denoted as *AccNet*.
- Operation of the WMN with internal traffic only is denoted as *Intern*.
- Operation of the network in a hybrid/mixed setup is denoted as *Mixed*.

To model these scenarios, we introduce three different classes of traffic: *Internet traffic (Inet)*, *symmetric traffic (Sym)* and *asymmetric traffic (Asym)*. In each *Inet* connection the MBS is a communication endpoint; we assume lower bandwidth for the uplink than for the downlink. *Sym* flows always request the same amount of bandwidth for both directions of a node pair, thus modeling, e.g. VoIP traffic. *Asym* traffic represents file transfers or video streaming sessions inside the network, with most of the demand in one direction and only a negligible amount in the reverse direction.

We instantiate the three modelled scenarios as shown in Table II. The traffic pattern for the *AccNet* scenario consists of *Inet* flows only. In the *Intern* scenario we combine the *sym* and *asym*[2] traffic pattern for communication among nodes of the WMN. The *Mixed* scenario combines all three types of traffic. Table II shows the individual setting for the traffic patterns for all studied scenarios, whereby the demand is uniformly distributed and specified as demand in minislots per frame per flow. The demands for the different WMNs scenarios were chosen such that the total demand for all flows in each scenario was in average around 80 minislots per frame. We assumed a total number of 67 available minislots, which corresponds to about 70% of all minislots in the ETSI(n = 8/7, 3.5 MHz, OFDM 256) mode of the IEEE 802.16 standard.

Fig. 5 and Table III show the results for MiH, He and HeNC for Setup II[3]. The results clearly indicate the superior performance of He vs. the baseline MiH if traffic patterns leave room for optimization.

TABLE II
TRAFFIC PATTERNS FOR SETUP II

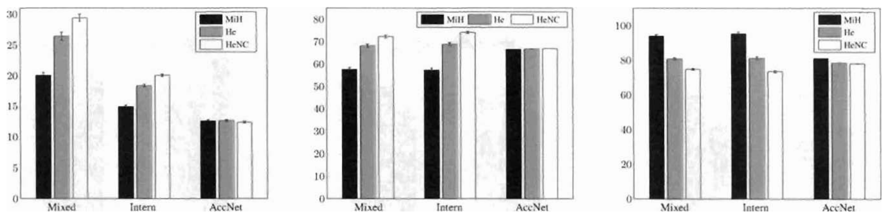
		Inet Up./Down.	Sym	Asym
AccNet	No. of flows	11 / 11	0	0
	Demand per flow	1-2 / 4-8	0	0
	Mean total demand	82.5 minislots per frame		
Intern	No. of flows	0	2x 8	6
	Demand per flow	0	3-4	3-5
	Mean total demand	80 minislots per frame		
Mixed	No. of flows	11 / 11	2x 5	5
	Demand per flow	1 / 1-3	3	2-5
	Mean total demand	80.5 minislots per frame		

[2]Asym. traffic is modelled with a reverse demand of zero, as the negligible demands are scheduled using short-term reservations not controlled by CORE, further for the reservation-based MAC a non-zero amount for the reverse demand has only low impact as soon as the reservation is issued.

[3]We omit the presentation of the results for the network coding variant MiHNC, because the inherent limitations in identifying NC sessions leads only to marginal performance gains relative to MiH.

TABLE III
SEARCH EFFORT, SUCCESSFUL REROUTES AND ESTABLISHED NETWORK CODING SESSIONS FOR SETUP II

	Combinations Tried			Reroutes			NC Sessions		
	Mixed	Intern	AccNet	Mixed	Intern	AccNet	Mixed	Intern	AccNet
MiH	37 ± 0	22 ± 0	22 ± 0	0	0	0	0	0	0
He	912 ± 6	531 ± 6	595 ± 4	11.0 ± 0.4	6.8 ± 0.2	3.2 ± 0.1	0	0	0
HeNC	870 ± 8	500 ± 5	578 ± 3	12.5 ± 0.4	7.5 ± 0.3	3.2 ± 0.1	14.0 ± 0.3	13.3 ± 0.3	1.9 ± 0.2



(a) Number of flows scheduled (b) Demand served (c) Blocked per demand

Fig. 5. Simulation results for Setup II

In particular, for the *Intern* scenario, He is able to admit around 20% additional flows compared to MiH, while HeNC realizes an improvement of about 33% in scheduled flows. For the *Mixed* scenario, the improvements are even more impressive: He outperforms MiH by scheduling around 30% additional flows, HeNC is able to increase the number of flows by nearly 50%. At the same time, Fig. 5 shows that *AccNet* does not provide this room for optimization, which is due to the bottleneck MBS. Since we assumed a total of 67 minislots, the MBS can serve a maximum demand of 67 minislots per frame, which also limits the performance of all three schemes as shown in Fig. 5 (b). Network coding does not help in this scenario either, because our heuristic prefers flows with high demand, i.e. downstream flows and does not permit sufficient upstream flows to the MBS to yield network coding opportunities. In contrast, as shown earlier, the possible gain further improves if network coding is enabled (HeNC) and sufficient network coding opportunities can be identified, which is true for the scenarios *Mixed* and *Intern*.

C. Summary of Results

The evaluation shows that CORE can very much improve the performance in WMNs. In networks that are tractable, our devised heuristics perform slightly worse than optimal solutions. However, please note that the former have been tuned such that they are able to operate in near real-time, while the latter are infeasible for realistic scenarios because of their runtime. Our scheme shows an excellent performance in realistic environments. In particular, the He as well as the HeNC variant significantly outperform the baseline of minimum-hop routing (MiH) by up to 50% improvement in terms of scheduled flows. At the same time, our algorithms are able to identify network coding opportunities and are practical to deploy network coding in TDMA/TDD mesh networks, even if these networks deploy per-hop link-level encryption.

VI. CONCLUSION

We see that CORE is able outperform the baseline schemes by admitting up to 50% of additional traffic. Further, to reach the solution CORE needs to search a very small fraction of the solution space. The distributed deployment of the solutions make the WMN robust to failures of the central server. In future we plan to investigate further optimizations to our heuristics, e.g. distributed computations of the heuristics.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.
[2] I. F. Akyildiz, X. Wang, and W. Wang. Wireless Mesh Networks: A Survey. *Computer Networks*, 47(4):445–487, March 2005.
[3] R. Barr. JiST/SWANS. http://jist.ece.cornell.edu/docs.html, 2004.
[4] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu. Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode. In *MobiHoc '05*, pages 78–89, 2005.
[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM '07*, 2007.
[6] I. Cidon and S. Moshe. Distributed Assignment Algorithms for Multihop Packet Radio Networks. *IEEE Transactions on Computers*, 38(10):1353–1361, October 1989.
[7] N. d'Heureuse. Cross-Layer Bandwidth Optimization Scheme for IEEE 802.16 Wireless Mesh Networks. Master's thesis, Technische Universitaet Darmstadt, 2007.
[8] GLPK. http://www.gnu.org/software/glpk/glpk.html.
[9] IEEE. IEEE 802.11 Standard. IEEE Std. 802.11, January 1999.
[10] IEEE. IEEE 802.16 Standard. IEEE Std. 802.16-2004, October 2004.
[11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. *ACM SIGCOMM CCR*, 36(4):243–254, October 2006.
[12] X. Lin and S. Rasool. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks. In *IEEE INFOCOM 2007*, pages 1118–1126, May 2007.
[13] P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz. A Case for Joint Near-optimal Scheduling and Routing in TDMA-basedWireless Mesh Networks: A Cross-layer Approach with Network Coding Support. In *IEEE MASS 07*, October 2007.
[14] P. S. Mogre, M. Hollick, and R. Steinmetz. The IEEE 802.16-2004 MeSH Mode Explained, KOM-TR-2006-08. Technical report, KOM, TU Darmstadt, Germany, ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2006-08.pdf, 2006.
[15] Y. E. Sagduyu and A. Ephremides. Joint Scheduling and Wireless Network Coding. In *NetCod 2005*, 2005.
[16] S. Sengupta, S. Rayanchu, and S. Banerjee. An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing. In *IEEE INFOCOM 07*, 2007.
[17] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas. Interference-aware IEEE 802.16 WiMax Mesh Networks. In *IEEE VTC 2005-Spring*, pages 3102–3106, June 2005.