Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.0000

CORE: Centrally Optimized Routing Extensions for Efficient Bandwidth Management and Network Coding in the IEEE 802.16 MeSH Mode[‡]

Parag S. Mogre[1*] and Nico d'Heureuse[2] and Matthias Hollick[1] and Ralf Steinmetz[1]

[1] *Multimedia Communications Lab, TU Darmstadt, Rundeturmstrasse 10, D-64283 Darmstadt, Germany*
[2] *NEC Europe Ltd., NEC Laboratories Europe, Kurfuersten-Anlage 36, D-69115 Heidelberg, Germany*

Summary

The IEEE 802.16 standard (WiMAX) specifies a MeSH 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 mode 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. We also see that CORE successfully increases the number of network coding sessions which can be established in the WMN. Further, the results provide insights into limiting factors for the gains which can be obtained in different typical usage scenarios for WMNs.
Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Wireless Mesh Networks, Network Coding, Routing, Scheduling, IEEE 802.16 Mesh Mode

1. Introduction

Wireless Mesh Networks (WMNs) are increasingly finding application in diverse application domains as can also be seen from the vigourous efforts in the standardization of wireless technologies and protocols for WMNs. Some examples of such standards for supporting next-generation WMNs are the IEEE 802.16 MeSH[†] mode, the ongoing IEEE 802.11s standardization, and the WirelessHART standard for setting up industrial wireless mesh/sensor networks. Each of these standards is tailored to meet different needs. The MeSH mode permits the setup of WMNs capable of supporting carrier-grade Quality of Service

[*]Correspondence to: Multimedia Communications Lab, Technische Universität Darmstadt, Rundeturmstrasse 10, D-64283 Darmstadt, Germany. E-mail: parag.mogre@kom.tu-darmstadt.de
[‡]This paper is an extended version of the research presented in our paper originally presented at the IEEE LCN 2008 [1].

[†]Throughout this paper we use the notation "MeSH" when referring to the mesh mode of operation of the IEEE 802.16 standard [2]

(QoS). The IEEE 802.11s standard aims at supporting WMNs much more efficiently than the contemporary IEEE 802.11 [3] standard. The WirelessHART [4] standard supports the setup of self-healing and self-organizing mesh topologies for industrial wireless networks, e.g. for monitoring production plants, etc. A unifying factor in these next-generation standards is the support for TDMA-based bandwidth reservation schemes for supporting stringent QoS requirements. This has opened up completely new application areas for WMNs which were previously not within the application domain for WMNs (a detailed survey on WMNs can be found in [5]).

In this paper we focus on the IEEE 802.16 MeSH mode, choosing it as a prototype for next-generation WMNs providing hard QoS support via bandwidth reservations. The MeSH mode is attractive for network providers wanting to extend the coverage of their existing networks in a flexible manner while simultaneously supporting carrier-grade QoS. The MeSH mode represents a paradigm shift in wireless medium access when compared to the contemporary IEEE 802.11 standard. The MeSH mode specifies extensive mechanisms for explicit reservation of bandwidth for transmissions on individual links in the WMN. It allows the provision of QoS on a packet-by-packet basis, where the per-hop handling for each packet is determined by the QoS field in the packet header (see [2, 6] for details of the standard specifications, and packet types/formats for the MeSH mode). Provision of end-to-end QoS is not within the scope of the MeSH mode specifications. From the point of view of the network operator deploying a WMN using the MeSH mode, the provision of carrier-grade QoS is, however, not the sole aim. The network operator normally wishes to increase the amount of flows/traffic admitted in the WMN as it usually means more revenue for the operator and efficient bandwidth utilization. Network coding [7] is a promising technique to achieve the latter goal in WMNs (e.g. [8]).

However, most of the work on network coding in WMNs has been carried out in the context of WMNs using the IEEE 802.11 standard (e.g. [9, 8]). Given the radical difference in access to the medium using the IEEE 802.16 MAC as compared to the IEEE 802.11 MAC, network coding solutions originally designed for and deployed in IEEE 802.11 networks need to be fundamentally reconsidered. We outline in Sec. 3 why the network coding solutions found commonly in the literature are highly inefficient if used in the MeSH mode and why a radically different approach is needed for gainful network coding in the MeSH mode.

Furthermore, an optimal deployment of network coding in the MeSH mode requires the joint optimization of 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. [9, 10]). However, the prior approaches found in the literature are not applicable to the MeSH mode or have computationally prohibitive costs to be of practical use in realistic traffic scenarios (Sec. 2.1 provides an overview of the MeSH mode, Sec. 2.2 discusses the related work).

In the MeSH mode we want to route packets on routes satisfying the QoS requirements of the flows. In particular, the packets should be routed along a single path to avoid jitter and reordering problems arising when multipath routing is used. Sufficient bandwidth (an integer number of minislots—the smallest unit of bandwidth allocation in the MeSH mode) needs to be reserved for the transmission on the individual links on the route. 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. CORE is designed to operate in dynamic WMNs with changing traffic demands. We presented a proof of concept for the CORE framework in [11]. 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 the underlying design principles for CORE which draw on the insights obtained from our previous work ([12, 13]).
- In Sec. 4 we present the CORE framework, which jointly optimizes the QoS aware routing, transmission schedule, and bandwidth savings via network coding in WMNs. CORE builds on standard routing protocols, extending their functionality (hence the name “Routing Extensions”) by enabling the adaptation of the routing tables at individual nodes in the WMN to achieve CORE’s goals.
- CORE uses heuristics (see Sec. 5) to achieve its optimization goal. CORE is designed such that the network operator can parametrize CORE’s heuristics for limiting 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 avoids a central point of failure, such that the WMN is able to continue with normal operation even if the central server running CORE’s heuristics fails. Further, 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.
- In Sec. 6 we evaluate the quality of the solutions computed by CORE 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 both the number of flows admitted in the WMN as well as the number of network coding sessions established, while requiring only 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.

Finally, in Sec. 7 we summarize the key contributions of this paper and give pointers to future research in the area.

2. 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.

2.1. MeSH Mode Background

The IEEE 802.16 MeSH mode [2] specifies the Medium Access Control (MAC) and the Physical (PHY) layers to enable the deployment of WMNs. The standard specifies the framework for medium access and bandwidth reservation. The algorithms for bandwidth reservation are, however, left open for optimization by individual vendors. 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 managed in distributed fashion by the individual nodes (Subscriber Stations, (SS)) using the distributed mesh election algorithm specified by the standard (see [2, 6, 14]).

Reservation of bandwidth for transmission of data messages in the data subframe can also be either 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. The scheduling tree does not need to contain all the nodes (SSs) in the WMN. With centralized scheduling bandwidth requests of individual SSs are propagated up the scheduling tree by the SSs to the MBS. The MBS computes the slots to be allocated to the individual nodes and transmits these as grants to its children in the scheduling tree. The grants are then propagated down the tree till all SSs in the tree receive their transmission schedules.

Distributed scheduling is more flexible and can be used to schedule transmissions on all the links (both those in and outside the scheduling tree) in the WMN. Using distributed scheduling a SS negotiates its transmission schedule via a three-way handshake with the neighbouring node which is to receive the transmission (see Fig. 1). Given the limitations of

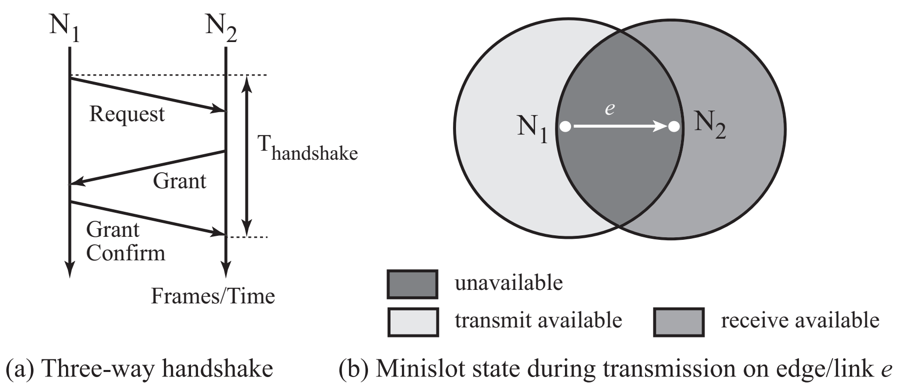


Fig. 1. Overview of three-way handshake for distributed scheduling and slot status update in the MeSH mode

centralized scheduling, without loss of generality, we assume that only distributed scheduling is used for the following discussion.

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 the 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; whereas per the standard's specification $\Delta F \in$ {1,2,4,8,32,128,∞}. 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. Fig. 2 visualizes the scope and validity of bandwidth reservations in the MeSH mode using distributed scheduling. Reservations are essentially rectangular areas in the two dimensional space defined by the minislot and the frames axes. 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 nor 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*. We assume that all the slot states of the nodes and their neighbours are in state *av* at the beginning of the handshake. Neighbours of the receiver (N_2) overhear the grant for link *e* and update the state for the granted slots to reflect that they may not transmit in the granted slots, since this would cause a collision at N_2. 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. However, there are subtle differences. In IEEE 802.11 networks each node hearing either the RTS or the CTS will withhold its own transmissions for the specified time duration. With the three-way handshake as specified in the IEEE 802.16 MeSH mode this is not the case. Referring to the example discussed earlier (see Fig. 1 (b)), nodes hearing the grant message will mark the slots as receive available, i.e. they may schedule data receptions in parallel but will not transmit themselves. On the other hand, nodes hearing the grant confirmation message will mark the slots as possibly transmit available and will be able to schedule their own transmissions in parallel to the transmission being scheduled by the three-way handshake. The other major difference between the three-way handshake and the RTS/CTS scheme is due to the fact that the control messages for the three-way handshake may be transmitted by the nodes in the MeSH mode only in slots in the control subframe which have been

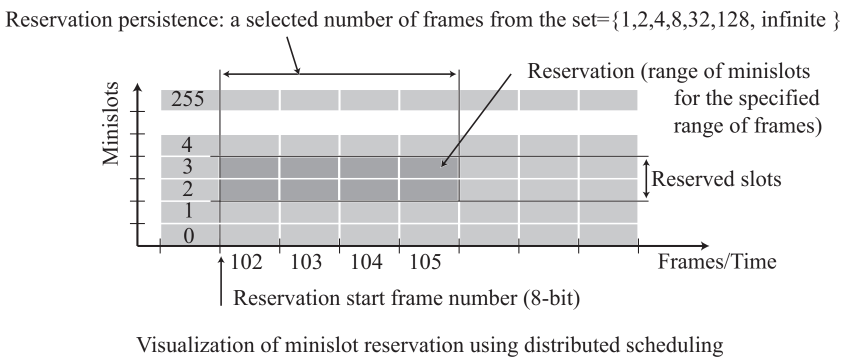

Reservation persistence: a selected number of frames from the set={1,2,4,8,32,128, infinite }
Reservation (range of minislots for the specified range of frames)
Minislots
255
4
3
2
1
0
Reserved slots
102
103
104
105
Frames/Time
Reservation start frame number (8-bit)
Visualization of minislot reservation using distributed scheduling

Fig. 2. Overview of slot reservations via distributed scheduling in the MeSH mode

won by the respective nodes via the mesh election process specified in the standard. Thus, it is usually not possible for a node to start the three-way handshake as soon as it has some data for transmission. Cao et al. have analyzed the three-way handshake delay in a very exhaustive manner in [14]. This also leads to the fact that unlike the RTS/CTS scheme the grant and grant confirm message as well as the request message might not be transmitted immediately after each other, which in turn opens the possibility for some other three-way handshake running in parallel in the neighbourhood to block the slots being granted such that the given handshake may fail. The other difference between the superficially similar looking mechanisms is the more flexible bandwidth reservation permitted by the three-way handshake in the MeSH mode. The bandwidth (slots) reserved via the three-way handshake in the MeSH mode need not lie temporally immediately after the handshake ends, but may be some frames into the future. Also, it is possible to reserve different and multiple regions of the frame $\times$ minislot (see Fig. 2) space via a single three-way handshake. Thus, in summary the MeSH mode permits more flexible mechanisms for bandwidth reservation as well as is more aggressive in spatial reuse of slots permitted in comparison to the similar looking RTS/CTS mechanism used by IEEE 802.11 networks.

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).

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

Thus, the interference model we use is basically given by the scheduling constraints specified in the IEEE 802.16 standard, and is similar to the protocol interference model specified in the literature (e.g. [15]). When talking about interference costs for the rest of the paper we will use the definition discussed earlier and used in Eqn. (1). Readers unfamiliar with the MeSH mode and interested in knowing more details may find a detailed overview in Ref. [6].

2.2. Related Work

Our work is inspired by the seminal work of Katti et al. [8]. They demonstrate the substantial benefits which can be achieved by using a simple form of network coding [7] in WMNs. We explain the basic working of network coding (as used in [8]) using Fig. 3. Using network coding, nodes do not rely on a simple store-and-forward operation when routing packets along a multihop path. Instead, nodes may

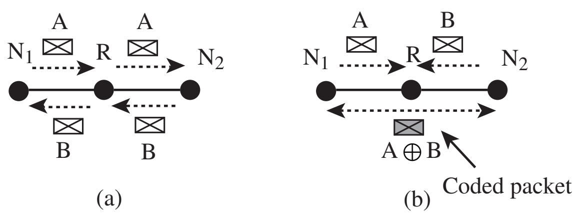


Fig. 3. Wireless network coding

combine information (using a chosen coding function) from incoming packets to form novel coded outgoing packets. These coded packets are then decoded at the neighbouring nodes receiving these, at some nodes down the path to the destination, or the final destinations[‡]. Consider Fig. 3; we have in the example the network topology as shown, with packets from node N_1 to N_2 and vice versa being relayed by node R which is a common neighbour of both nodes N_1 and N_2. Fig. 3 (a) shows the normal operation in a WMN where packets are stored at the intermediate relaying node R before being forwarded to the next hop. Consider that node N_1 wants to send a single packet, say packet A to node N_2, and node N_2 also wants to transmit a single packet to node N_1, say packet B. As show in Fig. 3 (a) using a normal store-and-forward operation in the WMN, four transmissions are needed for the exchange of the packets. Consider the same packets to be exchanged with network coding. In this case, when N_1 transmits the packet A, it also locally caches the transmitted packet for a certain duration of time after the transmission. Similarly, node N_2 caches packet B locally after transmitting it. Fig. 3 (b) shows the same exchange using network coding. In this case the relay R receives the uncoded packets A and B from nodes N_1 and N_2 respectively. It can then code the packets A and B together by using a XOR coding operation to form the coded packet $A \oplus B$. The relay node then transmits the coded packet in a single transmission to both the neighbouring nodes N_1 and N_2. These can then use the locally cached packets and the received coded packet and decode and correctly receive the uncoded packets meant for themselves. E.g. at node N_1 the operation $A \oplus (A \oplus B)$ is carried out to get packet B. Similarly, the packet A can be correctly decoded at the node N_2. Thus, as seen from Fig. 3 (b), only three transmission slots on the wireless medium are needed when using network coding in this example, as compared to the four transmission slots which were needed using a store-and-forward operation mode shown in Fig. 3 (a). Thus, even from this trivial example it is clear that significant bandwidth savings are possible when network coding is employed in the WMN.

Ref. [9] builds on [8] by considering network coding and routing as a joint problem. The authors in [9] use a linear programming based approach to find an optimal solution in an 802.11 based MAC. However, the solution presented requires 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 [16] for the details) as an ILP using the *GNU Linear Programming Toolkit* [17]. 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) the solver was not able to find a solution in reasonable time (24 hours); an increase in flows and/or links had similar effects on the solution time.

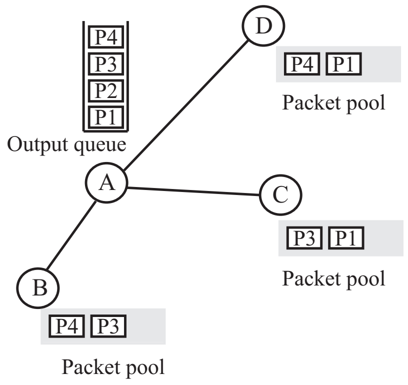
Almost all of the work in the field of network coding assumes a 802.11 or similar MAC. No study is made for a reservation based MAC like the MeSH mode. An important issue when using network coding is that it should not add a high delay penalty for the packets when coding is used (see [8] for arguments). 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 (see Sec. 3 for a more detailed discussion). 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 [14]), especially for multicast reservations.

[‡]Which nodes decode the coded packet depends on the coding scheme deployed. For practical reasons similar to those outlined in Ref. [8], we assume that the coded packets transmitted by a node should be decoded at the intended neighbours receiving these packets.

In the poster [11] 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 [11, 1] and studies the performance of CORE's heuristics in greater detail. Most of the other literature on scheduling and network coding is either restricted to multicast traffic (e.g. [10]), or does not permit spatial reuse for scheduling as specified for the MeSH mode (e.g. [18]). 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. It has been designed for reservation based WMNs using TDMA/TDD and implemented for the special case of the MeSH mode. The solution we present is deployed in a distributed fashion allowing the network to be resilient to failures of the centralized optimization entity.

3. Design Considerations for Network Coding in the MeSH Mode and Implications for CORE

Most of the practical work deploying NC ([19, 7] in WMNs is based on the seminal work of Katti et al. [8]. The work [8] uses the concept of opportunistic listening and opportunistic coding (using an XOR function to mix packets) for deploying NC in WMNs.



Packet to be transmitted at node A	Next hop for packet
P1	B
P2	D
P3	D
P4	C

Fig. 4. Opportunistic listening and coding

We explain the latter concepts using the example in Fig. 4. Assume we have the depicted state in the network and that the nodes temporarily store previously transmitted packets as well as packets overheard (via so called opportunistic listening) from neighbouring transmissions in a local packet pool. Assume that node *A* knows the lists of packets stored at each of its neighbours (e.g. via reception reports), and it has the shown packets in its transmission queue, with the next hops as shown in the table given in Fig. 4. Based on the available information, node *A* has different choices of which packets to code together via XOR before transmission. E.g. it can decide to code packets P3 and P1 and transmit P3⊕P1, or it can code and transmit the combination P4⊕P3⊕P1. The latter is of course the better choice for packets to combine as in this case nodes *B*, *C*, and *D* each are able to decode correctly and receive packets meant for transmission to them. Thus, the decisions as to which packets to code together is done at nodes in the network in an opportunistic manner, using the currently available packets in the node's queue, and information about the packets available with the node's neighbours.

The above approach is well suited for conventional IEEE 802.11 based WMNs which do not use bandwidth reservations. It is, however, extremely inefficient in the MeSH mode, and in general for reservation based WMNs as it involves a lot of overhead due to bandwidth reservation irrespective of the WMN standard being used. E.g. let us assume that node *A* decides to code the packets locally available as: P4⊕P3⊕P1. This combination should then be transmitted such that it is received at all the neighbours of *A* (nodes *B*, *C*, and *D*). In the MeSH mode this means that node *A* needs to have bandwidth reserved a priori for transmission to the set of neighbours in question before it can transmit the coded packet. The MeSH mode permits reservation of bandwidth only for transmission on individual links in the WMN according to the definitions in the standard. Thus, without amending the standard, it is not possible to reserve bandwidth for transmission to multiple nodes simultaneously. This implies that the node wanting to transmit the coded data to multiple neighbours simultaneously needs to perform a three-way handshake with each of the neighbouring nodes.

We have seen from the above discussion that coded packets need to be received by a given set of neighbours. Furthermore, in case opportunistic listening is deployed, then the non-coded packet transmissions also need to be received by a set of neighbouring nodes if these packets need to be used later for coding by the relay. In the MeSH mode as no transmission can take place without explicit bandwidth reservation it means that multicast bandwidth reservations are needed for both the coded transmissions as well as the uncoded transmissions in case of opportunistic listening. In prior work (see Refs. [12, 13] for details) we have shown analytically that these multicast reservations are very costly in terms of negotiation overhead as well as the delay incurred thereby. It is seen that these costs rise extremely with both an increase in the number of slots needed to be reserved in a frame, as well as with an increase in the number of neighbours to whom the transmission needs to be scheduled. We have thus seen that the network coding approach as in [8] is very inefficient for reservation based WMNs, and one should not make network coding decisions based on individual packets in the output queue as seen in the example discussed previously and depicted in Fig. 4.

We have shown in prior work ([12, 13]) that in reservation based WMNs like the MeSH mode the overhead of the three-way handshake (bandwidth reservation overhead) should be amortized by using the same coding decision over a stream of packets where possible. Here, network coding is no longer packet oriented but flow oriented, whereby packets belonging to a set of flows are regularly chosen for coding together and transmitted to a fixed set of receiving nodes. This permits the handshake procedure to be carried out once to reserve multicast bandwidth for a larger number of frames.

To make this more clear let us refer to Fig. 3. In contrast to packet-by-packet network coding decisions suitable for non-reservation based WMNs, in WMNs like the MeSH mode, we propose that the nodes observe the data arrival statistics of traffic packets from neighbouring nodes for relaying to other neighbouring nodes. This should then serve as a basis for initiating reservations for network coding and then the coding of data over packet streams itself. Thus in the example in Fig. 3, the node R would look at the mean arrival rate of packets from node N_1 for relaying to node N_2 and vice versa. If the data rate is above a chosen threshold then theoretically it would be possible for node R to act as a network coding relay and code minimum(F_1,F_2) bits together without adding delays. Here F_1 and F_2 are the data arrival rates for the two cross-flows in terms of bits per frame. Thus, if the node R observes that the mean data arrival rates of the flows are sufficient and that the flows tend to exist for a relatively long period of time to allow the amortization of the bandwidth reservation overhead for network coding, then the node R will initiate a multicast reservation handshake to reserve bandwidth for transmission to the nodes N_1 and N_2. Individual packets belonging to these streams will then be coded together similar to the coding for the individual packets A and B in Fig. 3. However, the bandwidth reservation handshake needs to be only performed at the start reserving multicast bandwidth for a large range of frames.

The exact approach for coding packets together and for reserving bandwidth for network coding are out of scope of this paper and CORE's functionality though. Details of the same may be found in [12, 13, 20, 21]. As CORE is designed for reservation based WMNs we will assume that such distributed mechanisms for network coding, which make network coding decisions looking at entire packet streams, as well as suitable mechanisms for reserving the required bandwidth are present in the WMN.

An additional aspect which should not be ignored is the presence of per link data encryption mechanisms in the MeSH mode. This implies that uncoded packets transmitted will be received correctly only at the intended recipients, and other neighbours overhearing these packets will not be able to interpret the contents of these packets. This excludes network coding solutions which assume opportunistic listening from being applied in such WMNs. Hence, for CORE, we will focus on optimizing and increasing the network coding opportunities similar to the Alice–Relay–Bob example in Fig. 3. This however does not exclude the operation of network coding sessions using opportunistic listening, these are however, not explicitly the goal of optimization for CORE.

CORE is designed using the insights obtained from the discussion above. We extended the MAC layer of the MeSH mode such that it supports network coding, additionally we also extended the protocol stack with a cross-layer interface allowing CORE to obtain information from the MAC layer, as well as provide information to the MAC layer and the routing layers. Additionally we designed an extended handshake procedure for reserving bandwidth for multicast transmissions, as is needed for transmitting coded data. We refer to this extended handshake as NC-handshake. The NC-handshake allows efficient negotiation of a common set of slots to be reserved for the multicast transmission. Details of the NC-handshake can be found in [13]. For the rest of the discussion we assume the presence of such an extended MAC layer for the MeSH mode.

We next give an overview of CORE's operation in Sec. 4 followed by the detailed operation in Sec. 5.

4. CORE: Functional Overview

CORE has been designed to help the nodes in the WMN to 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 by employing network coding.

CORE uses a central server to run its optimization heuristics (see Sec. 5) . Without loss of generality, for this paper we assume that the central CORE server is also the MBS and hence 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 next explain CORE's working in a nutshell with the help of an example.

4.1. CORE Operation Explained

Consider the toy-topology in Fig. 5 and two flows between the source destination pairs (*S*,*R*) and (*R*,*P*). Assume that initially only Flow 1: (*S*,*R*) exists. Also assume that Flow 1 uses route $S{\rightarrow}T{\rightarrow}R$. Now, when Flow 2: (*R*,*P*) enters the network, the source (*R*) estimates the mean data rate for the flow and notifies the MBS (for instance 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 $R{\rightarrow}Q{\rightarrow}P$ for Flow 2.

For this flow constellation, no opportunistic listening and coding similar to that used in COPE [8] is possible in the MeSH mode due to the per-link encryption at the MAC layer. The MBS computes an optimal route combination for the flows using the

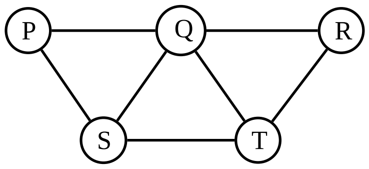


Fig. 5. Toy scenario to explain CORE's operation

algorithms mentioned in Sec. 5. It may, for example, determine that routing the flows as Flow 1 along: $S{\rightarrow}T{\rightarrow}R$ and Flow 2 along: $R{\rightarrow}T{\rightarrow}S{\rightarrow}P$ is a better solution since it generates opportunities for deploying network coding (e.g. at node *T*) in the network, does not violate the QoS requirements and, at the same time, minimizes interference in the network. The MBS sends control messages to the affected nodes notifying them of the required routing table updates and the schedule (bandwidth reservation) changes. In this example, the MBS informs nodes *R*, *T*, and *S* about the new routes. The nodes then change their individual routing tables accordingly. The MBS also notifies node *T* about the possibility for deploying network coding. However, 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 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 source nodes using the periodic network configuration messages transmitted in the control-subframe. The computations at the MBS are based on this traffic demand information. The MBS next 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. [16] provides details about the control protocols designed for CORE and the extensions we propose to amend to the IEEE 802.16 MeSH mode.

5. 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.

5.1. Route Preselection

The goal of the Route Preselection heuristic is to limit the number of routes per flow that are considered by the OptRC heuristic to an operator-defined value (κ), which limits the complexity of the optimization problem. 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.

5.2. Heuristic for the OptRC Subproblem

The solution to the OptRC problem uses the solution to the MaxSch problem (see 5.3) 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 or if the network capacity is insufficient, i.e. only an insufficient number of minislots is available. 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 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

$$\prod_{i=1}^{t} k_i \leq \Delta \tag{2}$$

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 an important role in the performance of Algorithm 1.

For the results in this paper (unless otherwise specified, e.g. for Sec. 6.3), 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 is placed before flows with lower traffic demands. Optimizing a larger number of flows in parallel is beneficial,

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 $2^{nd}$ 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 $f = \emptyset$
```

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

$$\begin{aligned}
\hat{C}(\kappa,0) &= 0\\
\hat{C}(\kappa,1) &= \kappa\\
\hat{C}(\kappa,2) &= \kappa^2 + 2\kappa\\
\hat{C}(\kappa,3) &= \kappa^3 + \kappa^2 + 3\kappa\\
\hat{C}(\kappa,4) &= \kappa^4 + 2\kappa^2 + 4\kappa\\
&\cdots\\
\hat{C}(\kappa,t) &= \kappa^t + \hat{C}(\kappa,\lceil t/2 \rceil) + \hat{C}(\kappa,\lfloor t/2 \rfloor). \qquad (3)
\end{aligned}$$

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.

5.3. 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 [22]. To find a maximal schedule we use an adapted version of the greedy maximal scheduling algorithm presented in [23] (similar scheduling algorithms may also be found in [22, 24]). The MaxSch heuristic assigns the first minislot to the link with the highest demand. Thereby, the corresponding set of conflicting 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 [8] is not possible. Hence, CORE focuses on optimizing for network coding when we have a traffic setup similar to the "Alice-Relay-Bob" scenario outlined in Ref. [8]. However, this does not limit the WMN from using network coding in other scenarios where possible.

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 links 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).

6. 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 [25]. For this paper, we restrict the discussion to the study of the network performance for the mesh topology shown in Fig. 6 for three distinct simulation setups. Additional experiments (different topologies, different flows) can be found in [16] 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. In Setup III we present an alternative flow sorting strategy for the *OptRC* algorithm which draws on the insights obtained from the results for Setup II. We investigate the influence of the new flow sorting strategy on the performance of CORE for the same settings as in Setup II. 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. Finally in Sec. 6.4 we look at some vital aspects of CORE in operation, in particular, we focus on how CORE is able to deploy the centrally computed solutions using default distributed components present in the WMN.

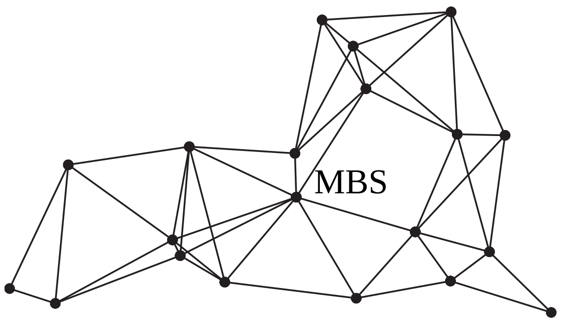


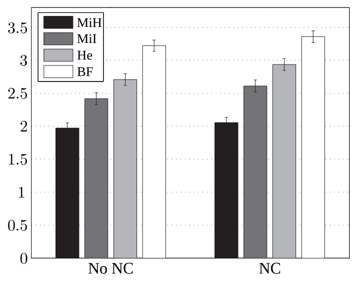
Fig. 6. Simulated 20 node WMN topology with Mesh Base Station (MBS)

6.1. Simulation Results: Setup I

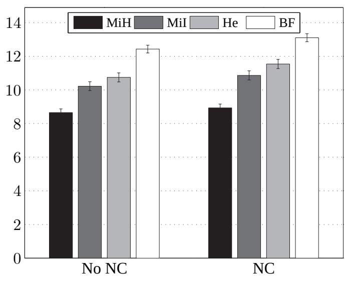
To find the globally optimal solution we implemented a brute-force algorithm which explores all feasible route combinations. Due to the prohibitive computational costs involved for the brute-force search we limited the study in Setup I to only 6 flows in the network with a limit of 20 minislots in the data-subframe. The traffic demands for the individual flows were uniformly distributed between 2–7 minislots per

Table I. Search Effort, Successful Reroutes and Established Network Coding Sessions for Setup I

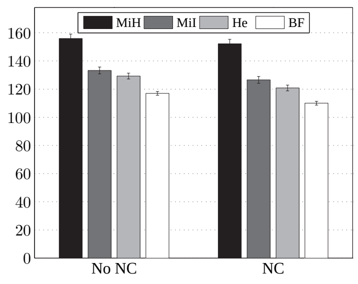
	Combinations Tried		Reroutes		NC Sessions	
	No NC	NC	No NC	NC	No NC	NC
MiH	6 ± 0	6 ± 0	0	0	0	0.38 ± 0.08
MiI	6 ± 0	6 ± 0	0	0	0	0.95 ± 0.14
He	74 ± 1	72 ± 1	0.845 ± 0.08	0.95 ± 0.09	0	1.49 ± 0.15
BF	15624 ± 0	15624 ± 0	1.15 ± 0.10	1.24 ± 0.10	0	1.83 ± 0.15



(a) Number of flows scheduled



(b) Demand served



(c) Blocked per demand

Fig. 7. Simulation results for Setup I

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 length). 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. 7(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. 7(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, 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. 7(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. 7(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. 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. 7. 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.

6.2. 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 source-destination 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*[§] 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. 8 and Table III show the results for MiH, He (CORE's heuristics with no network coding used) and HeNC (CORE's heuristics with network coding (NC) enabled) for Setup II[¶]. The results clearly indicate the superior performance of He vs. the baseline MiH if traffic patterns leave room for optimization.

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. 8 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. 8 (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*.

6.3. Simulation Results: Setup III and *Nearness*

The results in Sec. 6.2 show that for the AccNet scenario HeNC does not yield performance gains over the HE algorithm. Thus, the additional activation of network coding using CORE's HeNC heuristic was ineffective. This can be attributed to the default

[§]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.

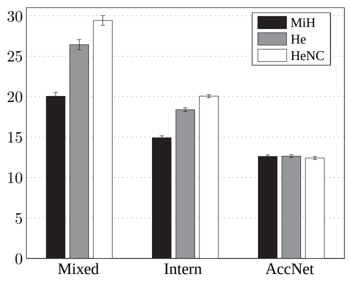
[¶]We omit the presentation of the results for the network coding variant MiHNC, because the inherent limitations in identifying NC sessions leads only to marginal performance gains relative to MiH.

Table II. Traffic Patterns for Setup II

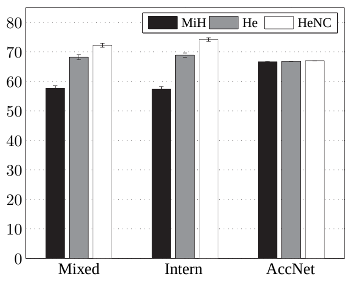
		Inet Up./Down.	Sym	Asym
AccNet	No. of flows	11 / 11	0	0
	Demand per flow	1-2 / 4-8	0	0
	Mean total demand	82.5 minislots per frame		
Intern	No. of flows	0	2x 8	6
	Demand per flow	0	3-4	3-5
	Mean total demand	80 minislots per frame		
Mixed	No. of flows	11 / 11	2x 5	5
	Demand per flow	1 / 1-3	3	2-5
	Mean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

Table III. Search Effort, Successful Reroutes and Established Network Coding Sessions for Setup II

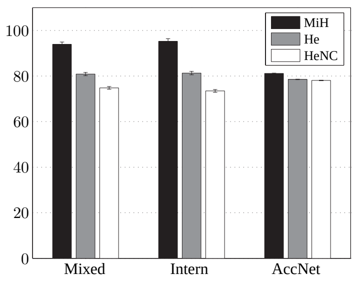
	Combinations Tried			Reroutes			NC Sessions		
	Mixed	Intern	AccNet	Mixed	Intern	AccNet	Mixed	Intern	AccNet
MiH	37 ± 0	22 ± 0	22 ± 0	0	0	0	0	0	0
He	912 ± 6	531 ± 6	595 ± 4	11.0 ± 0.4	6.8 ± 0.2	3.2 ± 0.1	0	0	0
HeNC	870 ± 8	500 ± 5	578 ± 3	12.5 ± 0.4	7.5 ± 0.3	3.2 ± 0.1	14.0 ± 0.3	13.3 ± 0.3	1.9 ± 0.2



(a) Number of flows scheduled



(b) Demand served



(c) Blocked per demand

Fig. 8. Simulation results for Setup II

ordering of flows based on their demand, which was used for the *OptRC* algorithm that is part of both heuristics. Due to this ordering of flows, the heuristics always chose to jointly optimize the routes for the higher throughput flows together before considering the flows with lower throughput. In the AccNet scenario, the flow setup was chosen to reflect the typical scenario in access networks, i.e. asymmetric Internet traffic with the volume of download traffic (traffic from the MBS towards the SSs) being much higher than the volume of upload traffic. Sorting the set of flows in the network in descending order of their demands results in the download flows to appear earlier in the sorted list, while the small upload flows appear at the end. Based on this sorting, CORE's heuristic jointly optimizes the routes for the downlink flows in parallel, but as these are all flows originating at the MBS (ingress/egress point for external traffic into/out of the WMN), the packets belonging to these flows are usually routed along a tree rooted at the MBS to the individual SSs. Thus, almost no latitude w.r.t. obtaining network coding opportunities by jointly optimizing the routes for the downlink flows exists. After having scheduled the high throughput flows using the central server, the share of the centrally managed bandwidth is nearly void. As a result the MBS can only process very few uplink flows of lower demand. The flows not fitting into the envelope assigned for centralized management are then left to be scheduled using the rest of the bandwidth in a distributed manner. Hence, for the studied AccNet scenario, `HeNC` does hardly setup any network coding sessions.

Obviously, if the system schedules and optimizes a pair of an uplink and its corresponding downlink flow such that the flows are routed along the same nodes (however in opposite directions) then a number of network coding opportunities can be created at the relay nodes (similar to that shown in Fig. 3). To identify and jointly combine such pairs of flows (not necessarily between the same source and destination nodes) we need to depart from the demand-based flow ordering. We propose a novel metric for determining the order of the flow, which we term as *nearness*. The intuitive idea behind nearness is to identify flows which are topologically close to one another in the WMN. Such flows are suitable candidate flows for optimizing their routes in parallel. To be more precise, the source node of one flow must be close (in the WMN topology) to the destination node of the second flow and vice versa. The most trivial case for this can be seen for the AccNet flows, where the source node of the uplink flow is the destination node for the downlink flow and the source node for the downlink flow is destination node for the uplink flow. We define the *nearness* of two flows as in Eq. (4).

$$\delta\left(f_x, f_y\right) = \frac{h\left(s_x, d_y\right) + h\left(s_y, d_x\right)}{2} \tag{4}$$

Where f_x and f_y are two flows with sources s_x and s_y and destinations d_x and d_y, respectively. $h(s,d)$ is the length of the minimum hop path from s to d. We can modify the OptRC Algorithm (Algorithm 1) to use the *nearness* metric when adding flows to a partition, by replacing lines 6–10 of Algorithm 1 by the following algorithm:

Algorithm 2 Flow selection using *nearness* heuristic

```
1: $x \leftarrow 0$
2: while $comb \cdot k_x \leq C$ do
3:     $comb \leftarrow comb \cdot k_x$
4:     $p \leftarrow p \cup \{f_x\}$
5:     $f \leftarrow f \setminus \{f_x\}$
6:     $x \leftarrow i$ such that $\delta\left(p_0, f_i\right)$ is minimal $\forall f_i \in f$
7: end while
```

As discussed in Sec. 5.2, p_0 is the most important flow in a partition. Instead of the flow with the highest demand when using the default metric, the *nearness* metric considers the flow that is nearest to all the other flows in its partition as p_0.

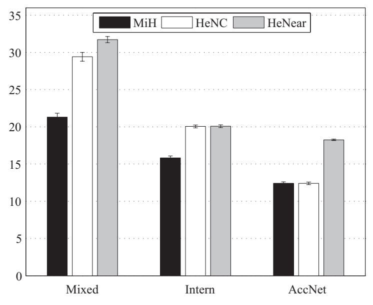
For the results presented in this subsection we compare the performance of the following algorithms `MiH`, `HeNC`, and `HeNear`. Where `HeNear` is CORE's heuristic with network coding enabled, and employing the *nearness* metric together with the correspondingly modified *OptRC* algorithm. Here, the main aim is to investigate the influence of the usage of the nearness metric on the capability of CORE to find and utilize network coding opportunities. For the results presented in Setup III, we use traffic settings similar to Setup II. We keep all other simulation parameters constant, except for the use of the nearness metric for `HeNear`. As both the heuristics are bound by the same computational costs, and there is no change to the computational costs incurred by the `MiH` algorithm, we refrain from presenting the computational costs for the simulations in Setup III.

Fig. 9 shows the results for `MiH`, `HeNc`, and `HeNear` for Setup III. As can be observed, the performance `HeNear` equals or betters the performance of `HeNC` in all the operating scenarios. Notably, significantly more flows are admitted into the network by `HeNear` in the *Mixed* and the *AccNet* scenarios. However, this does not translate to more total demand served. For the investigated setting we observe that `HeNear` schedules more lower demand flows which are *near* to some higher demand flows in the current partition of flows as compared to `HeNC` which prefers to first optimize the routes for all the higher demand flows in parallel before considering lower demand flows.

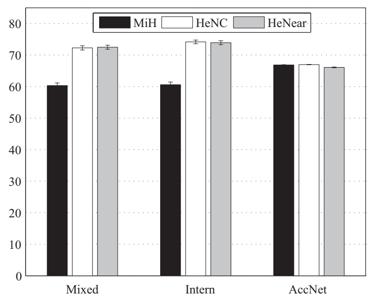
Fig. 10 shows the comparitive performance of `MiH`, `HeNC`, and `HeNear` considering their ability to setup network coding sessions in the WMN. As expected, `HeNear` is able to setup significantly more network coding sessions[‖] for the *Mixed* and *AccNet* scenarios. The gain in the number of network coding sessions setup is mainly due to the joint optimization of routes for traffic flows which form uplink/downlink pairs, but also other topologically close flows. However, the *nearness* metric is not necessarily always effective in setting up network coding sessions that further improve the network capacity. This can be seen from the slightly worse performance of `HeNear` when compared to `HeNC`. We can explain this result, because even if the source of one flow is close to the destination of another flow and vice versa, this does not imply that the individual nodes on the already established routes are close to each other. Rerouting the flows slightly thus does not necessarily allow the setup of significantly higher number of network coding sessions.

We observe a similar trend in the overall number of slots reserved (per frame) for the network coding sessions setup in the WMN. Although the number of network coding sessions setup in the *Mixed* scenario

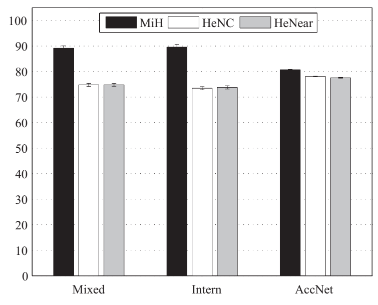
[‖]A network coding session consists of a node (relay) which transmits coded packets to its neighbours thereby relaying the packets via network coding instead of using simple store-and-forward



(a) Number of flows scheduled

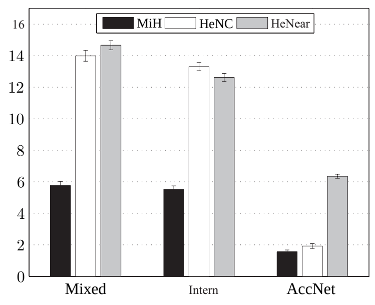


(b) Demand served

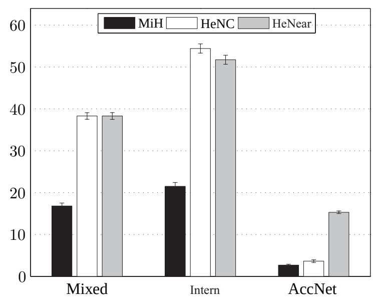


(c) Blocked per demand

Fig. 9. Simulation results for Setup III



(a) Number of NC sessions established



(b) Number of slots reserved for Multicast NC sessions

Fig. 10. Simulation results for Setup III

is slightly higher for `HeNear` as compared to `HeNC`, the total number of slots which are reserved for the network coding sessions is very close for both schemes. Again, this is possible, as `HeNC` tends to optimize the routes for the higher throughput flows in parallel first. Thus, it may e.g. setup a single NC session where the slots reserved for coding are 10 slots per frame. On the other hand, `HeNear` tends to optimize routes for nearby flows first, and may thus e.g. set up 10 NC sessions for 10 uplink/downlink flow pairs. Since the traffic may be asymmetric in volume for the latter case, flows are matched for network coding that e.g. need 1 slot per frame of uplink bandwidth, and each 10 slots per frame of downlink bandwidth. Hence due to the asymmetry in the traffic, at most one slot per frame will be attributed to network coding for each of the established NC sessions.

We can conclude that using the *nearness* metric one can very efficiently identify and select uplink/downlink or similar flow pairs for jointly optimizing the routes. This leads to a significantly increased number of network coding opportunities using `HeNear`, especially in the *AccNet* scenario. In contrast, in Setup II, `HeNC` was not able to show any significant performance gains over `He`. However, to achieve a significant capacity gain in the network, `HeNear` relies not only on the proposed flow sorting, but also needs to operate under network and traffic parameters such as topology restrictions and flow asymmetry enabling the necessary latitude for rerouting of traffic.

6.4. Simulation Results: Analysis of CORE In Operation

In the previous sections, i.e. Sec. 6.1, Sec. 6.2, and Sec. 6.3 we evaluated the quality of the heuristics computed by the CORE Server in different settings. However, as discussed, CORE relies on distributed components, i.e. components at each node in the WMN for the final deployment of network coding, as well as for the reservation of bandwidth and management of the transmission schedules. In this section we will look at selected results which demonstrate the critical aspects of CORE's working, and show that CORE is in fact able to approach the centrally computed solutions via distributed components.

We explained with a running example and toy topology shown in Fig. 5 the overall operation of CORE in a nutshell. Let us once again refer to Fig. 5, and assume the same network frame parameters for the IEEE 802.16 MeSH mode as specified in Sec. 6.2. Let us assume that we have the following flows in the network: Flow 1: from node P, to node R starting at time 10 sec and stopping at 20 sec with a constant data rate (demand) equivalent to 6 minislots per frame; and Flow 2: from node T to node P, starting at 13 sec and stopping at 17 sec with a demand equivalent to 11 minislots per frame.

Initially, the WMN will use the default routes provided by the routing in place in the network (i.e. in our case shortest hop paths). Thus, Flow 1 is routed via the path $P \rightarrow Q \rightarrow R$, which, incidentally is also the minimum interference path, and will provide the best delay when sufficient bandwidth is reserved all along the path. Thus, after the initial bandwidth requests have been processed (i.e. the control delay and overhead we discussed in depth in Sec. 3), the mean end-to-end delay for packets belonging to Flow 1 comes down to 20ms as shown in Fig. 11 at point ①.

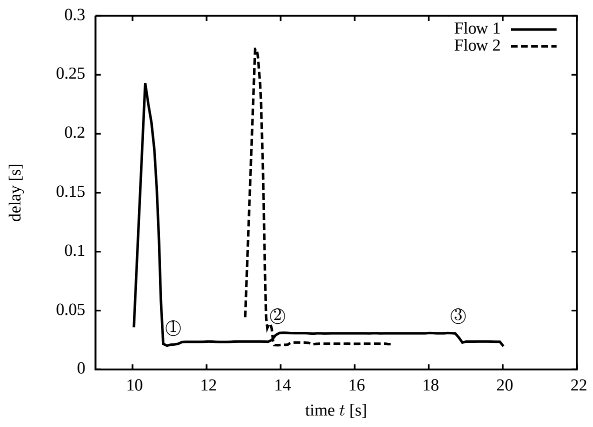


Fig. 11. End-to-end delays for the toy topology from Fig. 5

This is in our setup the lowest delay which would be achievable for this path assuming that data usually cannot be forwarded in the same frame in which it has been received by a node from the previous hop. The frame duration for the selected IEEE 802.16 parameters is 10ms. Now, when Flow 2 starts at time $t = 13s$, Flow 2, similar to Flow 1 will be routed along the default shortest hop path (here, route $T \rightarrow S \rightarrow P$).

The central CORE server, here node Q, is then informed about the demand of this flow too, and uses its heuristics to determine that the overall minimum interference route set uses route $P \rightarrow S \rightarrow T \rightarrow R$ for Flow 1 and $T \rightarrow S \rightarrow P$ for Flow 2.

This increases the path length for Flow 1 by one hop, however, the rerouting of the flows as per the routes computed by the heuristics at the central server reduce the overall interference in the network by allowing node S to act as a network coding relay for packets flowing in both directions via the node between the nodes P and T. As discussed in Sec. 4 the CORE server then sends appropriate control messages to the nodes in the WMN to instruct them to update their routing tables and use the routes proposed by the CORE server for the individual flows instead of the default routes. Thus, the CORE server basically provides an extended routing functionality adapting the default routes where needed, hence the name Centrally Optimized Routing Extensions.

However, in addition to notifying the individual nodes to adapt their routing tables, the CORE server also instructs the nodes to adapt their bandwidth reservations on the individual links in advance to the changes of the routing tables. It should be noted that besides these instructions which are provided via a cross-layer interface, CORE does not play any role in the actual bandwidth reservations nor does it reserve multicast bandwidth for network coding, or control network coding operations centrally, but the distributed bandwidth reservation mechanisms present on the node in the MeSH mode are used. Note, we assume that these default components only needed to be slightly modified to both enable interpretation of CORE's instructions, as well as to enable multicast bandwidth reservation and network coding functionality. Discussing the details of these mechanisms is not possible in the limited scope of this paper, which focuses on the overall operation of CORE, and its heuristics. The interested reader is however referred to Refs. [16, 20, 26, 21] for additional details about the bandwidth reservation as well as network coding solutions for reservation based WMNs.

Thus, based on the route updates received from the server, the nodes will reroute Flow 1 such that it now takes a longer route. This is reflected by the slight increase in the end-to-end delay of packets for this flow (see ② in Fig. 11). However, due to the additional advance bandwidth reservation change instructions sent by the CORE server, the transition from one path to the other is relatively smooth, and does not incur the comparatively high initial delays due to the time required for bandwidth reservation along the path. As Flow 2 ceases at $t = 17$s, Flow 1 is rerouted back to its default path (③ in Fig. 11). We used the toy-topology to be able to easily show the effect of the route change for flows, and how the CORE framework mechanisms allow a smooth transition from one path to another for the flows. For the time duration where both the flows are active in the network, and there exists a cross-flow at node S, this node may set up a network coding session allowing network coding to be deployed. The mechanisms for network coding itself are however beyond the influence of CORE, and there are different possible mechanisms to implement network coding. In general, as we have shown in our work Ref. [12, 13] in reservation based systems, network coding is meaningful only for long-lived flow pairs where the additional reservation overhead can be amortised by coding a large number of packets arriving over time for the chosen flow pair/set. We highlighted this aspect in brief in Sec. 3.

We now once again turn back to the bigger topology shown in Fig. 6. The aim is to see if CORE in operation does indeed bring the expected increase in terms of additional network coding sessions which are enabled in the WMN for a given set of flows in the WMN. We use the same IEEE 802.16 frame parameters as considered in Sec. 6.2, and look at the Mixed Traffic scenario specified in Tab. II. We again assume that the CORE server's *OptRC* heuristic sorts flows based on their demands. For this setup, based on the computation at the CORE server, we see from Fig. 10 (a) that CORE using the heuristics (i.e. `HeNC`) is expected to enable on an average 14 network coding sessions, which when compared to the around 6 network coding sessions expected to be enabled by the standard routing in the WMN is a significant improvement.

We simulated the operation of the network for a total of 11000 frames, with the Mixed Traffic scenario such that the individual flows started at random times between frame 2000 and 4000 and then existed for the rest of the simulation. We used a IEEE 802.16 MAC layer with distributed bandwidth reservation mechanisms (see Ref. [26]) with additional extensions permitting nodes to individually setup network coding sessions (i.e. act as relay for a network coding session) based on statistical analysis for the traffic relayed by the nodes for various flows to choose flow pairs for whom network coding would be fruitful (see [20, 21] for details). Exact details of the start times of the individual traffic flows can be found in [21]. The network coding mechanism at the nodes was such that it would start a network coding session and actively code packets belonging to a flow pair only when it observed that the flows have existed for a certain minimum duration, and have sufficient traffic demand to profitably gain from network coding.

Fig. 12 shows the number of network coding sessions established using the implemented network coding enabled MAC layer. Fig. 12 shows that the total number of network coding sessions established (i.e. for which the nodes acting as relays were able to observe cross-flows with sufficient traffic demand, and where these flow's existed long enough) in the WMN when using CORE is more than double the number of network coding sessions which could be established in the WMN for the same set of flows without CORE in operation. Furthermore one sees that with more and more flows entering the network (between frames 2000 and 4000) the number of possible network coding sessions increases for both the WMN with and without CORE. This is quite possible as some default routes may lead to suitable network coding opportunities. Additionally, one also sees that the network coding sessions are established with a certain lag to the flow's entering the network. This is due to the implementation we used where network coding sessions were established only once the flows had existed for a minimum time duration and were observed to have a minimum traffic demand. But, in both the cases, i.e. when operating with CORE enabled, and without CORE enabled, the MAC layer implementation was the same, so these effects are similar for both scenarios. One sees that the CORE server adapts the routes for the flows multiple times (i.e. with the entry of flows where it finds out that another route combination may lead to a schedule blocking less minislots, similar to the case seen for the toy-topology discussed earlier). This is seen by the steady step-wise increase in the number of network coding sessions which could be established.

Now, if one compares the number of network coding sessions which the WMN was able to establish with the number of network coding sessions which had been estimated by the computations at the CORE server for the scenario (see Fig. 10 (a)), we see that the distributed means of implementing CORE's centrally computed solutions come close to the central solution in terms of the network coding sessions which could be established. The difference to the central solution is only around two sessions in this simulation, and a MAC layer implementation which would be less conservative in establishing network coding sessions would further reduce this difference. The conservative nature of our MAC layer implementation is especially seen when we look at the number of network coding sessions which are established without CORE, here too the number of network coding sessions is slightly less than that which is theoretically possible. However, we have seen that in reservation based WMNs network coding involves considerable overhead for coding as well as maintenance and reservation of suitable schedules for transmitting packets involved in network coding. Additional results for the scenario can be found in Ref. [20, 21].

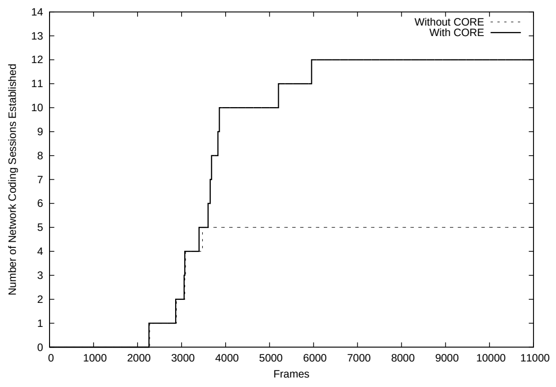


Fig. 12. Network coding sessions setup over time with and without CORE

6.5. 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. Further, the network operator can trade off more computational complexity for better solutions, if the additional computational resources and time are available in the WMN. CORE 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.

7. Conclusion

In this paper we outlined basic design principles which need to be considered when developing network coding solutions for the MeSH mode of IEEE 802.16, and TDMA/TDD based WMNs in general. CORE was designed using these insights. Our results show that CORE is able outperform the baseline schemes by admitting up to 50% of additional traffic. Further, to reach the optimized solution CORE needs to search only a very small fraction of the solution space, and can thus operate in near real-time in realistic scenarios. The distributed deployment of CORE's solutions make the WMN robust to failures of the central server. We see that in distributed operation CORE allows the WMN to establish significantly higher number of network coding sessions in the WMN, and approaches very close to the centrally computed solution. We can also conclude that in general sorting the flows in the order of their traffic volume for the *OptRC* algorithm leads to good performance. However, enhanced gains can in principle be obtained using the *nearness* metric for sorting flows in scenarios with a lot of traffic to and from the MBS. The network operator is provided with a myriad of choices for sorting the flows, e.g. based on their priority class, etc. and CORE by no way limits the possibilites. In future work we plan to look into this very important aspect in detail to derive insights for dynamically adapting the sorting scheme to best suit the operator specified goals. Another interesting aspect we plan to investigate is completely distributed solutions to the problem presented in this paper. Another focus of investigation we will be efficient and completely distributed solutions instead of the central optimization used by CORE.

8. Acknowledgements

The authors express their thanks to the anonymous reviewers whose constructive feedback and comments helped improve the quality of this manuscript a lot.

References

1. Mogre PS, d'Heureuse N, Hollick M, Steinmetz R. CORE: Centrally Optimized Routing Extensions for the IEEE 802.16 MeSH Mode. *IEEE LCN 2008*, 2008.
2. IEEE. 802.16 Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Std. 802.16-2004 October 2004.
3. IEEE Computer Society. 802.11 IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific requirement, Part 11: Wireless LAN

Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std. 802.11-2007 June 2007.
4. HART Communication Foundation. HART 7 Specification. Standard September 2007.
5. Akyildiz IF, Wang X, Wang W. Wireless Mesh Networks: A Survey. *Computer Networks* March 2005; **47**(4):445–487.
6. Mogre PS, Hollick M, Steinmetz R. 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.
7. Ahlswede R, Cai N, Li SR, Yeung RW. Network Information Flow. *IEEE Transactions on Information Theory* July 2000; **46**(4):1204–1216.
8. Katti S, Rahul H, Hu W, Katabi D, Medard M, Crowcroft J. XORs in the Air: Practical Wireless Network Coding. *ACM SIGCOMM Computer Communication Review* October 2006; **36**(4):243–254.
9. Sengupta S, Rayanchu S, Banerjee S. An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing. *IEEE INFOCOM 07*, 2007.
10. Sagduyu YE, Ephremides A. Joint Scheduling and Wireless Network Coding. *First Workshop on Network Coding, Theory, and Applications (NetCod 2005)*, 2005.
11. Mogre PS, d'Heureuse N, Hollick M, Steinmetz R. A Case for Joint Near-optimal Scheduling and Routing in TDMA-basedWireless Mesh Networks: A Cross-layer Approach with Network Coding Support. *IEEE MASS 07*, 2007.
12. Mogre PS, Hollick M, Kropff M, Steinmetz R, Schwingenschloegl C. A Note on Practical Deployment Issues for Network Coding in the IEEE 802.16 MeSH Mode. *IEEE WiNC 2008, IEEE SECON 2008*, 2008.
13. Mogre PS, Hollick M, Schwingenschloegl C, Ziller A, Steinmetz R. *WiMAX Evolution*, chap. WiMAX Mesh Architectures and Network Coding. Wiley-Interscience, 2009; 145–162.
14. Cao M, Ma W, Zhang Q, Wang X, Zhu W. Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode. *MobiHoc '05*, 2005; 78–89.
15. Gupta P, Kumar PR. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory* 2000; **46(2)**:388–404.
16. d'Heureuse N. Cross-Layer Bandwidth Optimization Scheme for IEEE 802.16 Wireless Mesh Networks. Master's Thesis, TU Darmstadt 2007.
17. GLPK. http://www.gnu.org/software/glpk/glpk.html.
18. Chachulski S, Jennings M, Katti S, Katabi D. Trading Structure for Randomness in Wireless Opportunistic Routing. *SIGCOMM '07*, 2007.
19. Fragouli C, Le Boudec J, Widmer J. Network coding: an instant primer. *SIGCOMM Comput. Commun. Rev.* 2006; **36**:63–68.
20. Wieber M. Optimierte Verfahren der Bandbreiten-Verwaltung: IEEE 802.16 Mesh-Modus mit erweiterter Unterstützung fr Network Coding. Master's Thesis, TU Darmstadt 2008.
21. Wowra J. Evaluation of Network Coding Opportunities in Wireless Mesh Networks, Bachelors Thesis, Technische Universitaet Darmstadt 2009.
22. Cidon I, Moshe S. Distributed Assignment Algorithms for Multihop Packet Radio Networks. *IEEE Transactions on Computers* October 1989; **38**(10):1353–1361.
23. Lin X, Rasool S. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks. *IEEE INFOCOM 2007*, 2007; 1118–1126.
24. Wei HY, Ganguly S, Izmailov R, Haas Z. Interference-aware IEEE 802.16 WiMax Mesh Networks. *IEEE VTC 2005-Spring*, 2005; 3102–3106.
25. Barr R. JiST. http://jist.ece.cornell.edu/docs.html 2004.
26. Mogre PS, Hollick M, Steinmetz R, Dadia V, Sengupta S. Distributed Bandwidth Reservation Strategies to Support Efficient Bandwidth Utilization and QoS on a Per-link Basis in IEEE 802.16 Mesh Networks. *IEEE LCN 2009*, 2009.

Authors' Biographies

Parag S. Mogre obtained the M.Tech. degree in Computer Science and Engineering from the Indian Institute of Technology, Guwahati in 2004. His Masters Thesis has won the KuVS prize in 2005. Since October 2004 he is active as a researcher at the Multimedia Communications Lab (KOM), TU Darmstadt, Germany, and is pursuing his Ph.D. there. His research focuses on medium access control, routing and resource management in wireless mesh and ad hoc networks. He has over 10 patent applications in the area of WiMAX based mesh networks, and has been awarded the invention achievement award for 2007/08 by the Dept. of Electrical Engineering and Information Technology, at the TU Darmstadt. He is leading the mobile communications and sensor networking group at KOM, TU Darmstadt.

Matthias Hollick is heading the Secure Mobile Networking lab at the Computer Science department of TU Darmstadt, Germany. He received his doctoral degree (Dr.-Ing.) in 2004 from the TU Darmstadt. His research focus is on secure and quality-of-service-aware communication for mobile and wireless ad hoc, mesh, and sensor networks. Dr. Hollick has been working and researching at TU Darmstadt, UC3M, and the UIUC. In 2005, Dr. Hollick has received the Adolf-Messer Foundation award.

Ralf Steinmetz is professor and chair of the Multimedia Communications Lab at the TU Darmstadt. Together with more than 30 researchers, he is working towards the vision of "seamless multimedia communications". He has contributed to over 400 refereed publications. He is an ICCC Governor, Fellow of both the IEEE and ACM. Professor Dr. Ralf Steinmetz is a member of the Scientific Council and president of the Board of Trustees of the international research institute IMDEA Networks.

Nico d'Heureuse obtained the M.Sc. degree in Electrical Engineering and Information Technology in 2007 from the TU Darmstadt, in Germany. He is currently a research associate at the NEC Europe Ltd., NEC Laboratories Europe, Networking Division, at Heidelberg, Germany. His current research interests include solutions for SPIT, and research in wireless mesh networks.