A Distributed Algorithm to Achieve Transparent Coexistence for a Secondary Multi-hop MIMO Network

Xu Yuan, *Member, IEEE,* Xiaoqi Qin, *Student Member, IEEE,* Feng Tian, *Member, IEEE,* Yi Shi, *Senior Member, IEEE,* Y. Thomas Hou, *Fellow, IEEE,* Wenjing Lou, *Fellow, IEEE,* Scott F. Midkiff, *Senior Member, IEEE,* and Sastry Kompella, *Senior Member, IEEE*

***Abstract*—The transparent coexistence (TC) paradigm allows simultaneous activation of the secondary users with the primary users as long as their interference to the primary users can be properly canceled. This paradigm has the potential to offer much more efficient spectrum sharing than traditional interweave paradigm. In this paper, we design a distributed algorithm to achieve this paradigm for a secondary multi-hop network. For interference cancelation (IC), we employ MIMO at secondary nodes. We present a distributed iterative algorithm to maximize each secondary session's throughput while meeting all IC requirements under TC. By maintaining two local sets for each node, we can keep track of the node's IC responsibility. Although no explicit node ordering is maintained in our distributed algorithm, we prove that our distributed data structure at each node (with the use of two local sets) can be mapped to an explicit global node ordering for IC among all nodes in the network. This guarantees that each active node's degree-of-freedoms (DoFs) allocated for IC is feasible at the physical (PHY) layer. Our algorithm is iterative in nature and all steps can be accomplished based on local information exchange among the neighboring nodes. We present simulation results to show that the performance of our distributed algorithm is highly competitive when compared to an upper bound solution from the corresponding centralized problem.**

***Index Terms*—Primary network, secondary network, spectrum sharing, coexistence, distributed algorithm, multi-hop network, MIMO, interference cancelation.**

I. Introduction

A spectrum sharing paradigm is defined by how the secondary and the primary users achieve coexistence. In [3], Goldsmith *et al.* outlined three main paradigms, namely *interweave, underlay,* and *overlay.* Interweave is a simple but conservative approach that follows the traditional interference avoidance paradigm. Under interweave, a secondary network is allowed to access radio spectrum only when it is not in conflict with the primary users in time, frequency, or space [2], [5], [17]. On the other hand, overlay is considered an aggressive spectrum sharing paradigm as it encourages proactive cooperation between the primary and secondary networks in data forwarding [7], [8], [11], [15], [23], [26]. In terms of spectrum sharing efficiency and network performance, overlay represents the ultimate coexistence paradigm, although its actual adaptation and deployment may still be years away due to the need of significant change in primary users' behavior. In this research, we focus on the underlay paradigm, which is considered as a major step forward beyond the interweave paradigm while requiring minimal change on the primary network. The underlay refers to that secondary users may be active simultaneously with the primary users in the same vicinity and in the same frequency, as long as the secondary user's interference to primary users are negligible (or below a given threshold).

Underlay coexistence paradigm has been explored in [1], [10], [24], [25]. In [1], Gao *et al.* studied the transmission strategies for a MIMO secondary link with a primary link. They proposed a secondary transmission strategy consisting of environment learning, channel training, and data transmission. In [24], Zhang and Liang studied the transmission strategy for a single secondary MIMO link coexisting with multiple primary receivers with interference-power constraints. In [25], Zhang *et al.* studied the secondary-link beamforming pattern to achieve the coexistence of a single secondary link with multiple primary links. They aimed to maximize the secondary user's throughput while keeping the interference temperature at the primary receivers below a certain threshold. In [10], Kim and Giannakis studied the coexistence of multiple secondary links with one primary link. They proposed a distributed resource allocation algorithm to maximize the weighted sum rate of secondary links under a transmit power constraint at the secondary transmitters and an interference power constraint at the primary receiver. All these prior efforts were from information theoretic perspective. A common limitation of these prior efforts is that they are all limited to very simple network settings, e.g., several nodes or link pairs, all for single-hop communications.

In a recent study [19], we explored the underlay paradigm for a secondary multi-hop network under the name of *transparent coexistence (TC).* Under TC, there is no change on the primary network's behavior. It uses the spectrum as it wishes

Manuscript received July 9, 2015; revised January 18, 2016; accepted May 27, 2016. An abridged version of this paper was presented at IEEE IPCCC, Austin, TX, USA, 2014 [21]. The associate editor coordinating the review of this paper and approving it for publication was W. Wang.

X. Yuan, X. Qin, Y.T. Hou, W. Lou, and S.F. Midkiff are with the Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA. (email: {xuy10, xiaoqi, thou, wjlou, midkiff}@vt.edu).

F. Tian is with the Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China. (e-mail:tianf@njupt.edu.cn).

Y. Shi is with the Intelligent Automation Inc., Rockville, MD, 20855, USA. (e-mail: yshi@vt.edu).

S. Kompella is with the U.S. Naval Research Laboratory, Washington, DC 20375, USA. (email: sastry.kompella@nrl.navy.mil).

and is not concerned with the needs of the secondary network. On the other hand, the secondary network is allowed to access the spectrum in the *same* time, frequency, and location with the primary network, as long as its activities are "invisible" to the primary network. Such transparency is achieved by having the secondary network proactively cancel its interference to the primary network with powerful physical (PHY) layer techniques so that the primary nodes do not feel the presence of the secondary nodes. As a result, simultaneous activation of the secondary network along with the primary network is possible. In [19], we developed centralized mathematical models to characterize (i) *inter-network* interference cancelation (IC) relationships between two networks – secondary transmitters need to cancel their interference to the primary receivers while secondary receivers need to cancel the interference from the primary transmitters; and (ii) *intra-network* IC – secondary nodes need to perform IC within their own network so that data can be transported successfully within the secondary network.

The results in [19] showed the concept of achieving TC for a multi-hop primary and secondary network through a centralized solution. But it is also desirable to have a distributed solution to achieve TC. The main contribution of this paper is the development of a distributed scheduling algorithm for the secondary network to achieve TC with the primary network, while maximizing its own network throughput. For IC, we assume each secondary node is equipped with MIMO, while there is no requirement on the primary nodes. We employ a MIMO IC model that was developed in [14] to keep track of degree-of-freedoms (DoFs) allocation for transporting data streams (i.e., spatial multiplexing (SM)) and IC. It was shown in [14] that this IC model is efficient in DoF allocation while guaranteeing feasibility in the final solution. By feasibility, we mean there exists a feasible precoding and decoding vector for each data stream at the PHY layer. However, this model is centralized in nature and requires to maintain a global node ordering among the secondary nodes in the network, which is not possible in a distributed network environment. In this paper, instead of maintaining a global node ordering, we only maintain two local sets at each node to keep track of the node's IC responsibilities. We show how to establish, maintain, and update these two local sets at each node in each iteration of our distributed algorithm. Our distributed algorithm increases the data stream on each active link iteratively based on local computation. Since the nodes in the two local sets of a node directly affect the node's IC responsibility, our algorithm attempts to switch nodes in the two sets if it can improve the IC structure. Although no explicit node ordering is maintained in our distributed algorithm, we prove that our distributed data structure at each node (with the use of two local sets) can be mapped to an explicit global node ordering for IC among all nodes in the network. From this global node ordering for IC among all nodes, we show there exist a set of feasible precoding vectors at each secondary transmitter and a feasible set of decoding vectors at each secondary receiver so that all data (in both primary and secondary networks) can be transported free of interference. Through numerical results, we show that the iterative distributed algorithm that we propose offers competitive performance when compared with an upper bound result from centralized optimization.

TABLE I
NOTATION

Primary Network	
$\mathcal{P}$	The set of nodes in the primary network
T	The number of time slot in a frame
$\tilde{\mathcal{F}}$	The set of sessions in the primary network
$\tilde{\mathcal{L}}$	The set of active primary links
Secondary Network	
$\mathcal{S}$	The set of nodes in the secondary network
S	The number of secondary nodes in the network, $S = \|\mathcal{S}\|$
A_i	The number of antennas at secondary node $i \in \mathcal{S}$
$\mathcal{F}$	The set of sessions in the secondary network
$\mathcal{L}$	The set of secondary links

The remainder of this paper is organized as follows. In Section II, we describe our problem. Section III presents the design of an iterative distributed algorithm to achieve TC for a secondary multi-hop network. In Section IV, we present a feasibility proof of our distributed algorithm at the PHY layer. In Section V, we analyze the complexity and overhead of our distributed algorithm. Section VI presents numerical results and demonstrates the competitive performance of the proposed distributed algorithm. Section VII concludes this paper.

II. Problem Description

In this paper, we consider a multi-hop primary network (with a set of nodes $\mathcal{P}$) and a multi-hop secondary network (with a set of nodes $\mathcal{S}$) that are co-located in the same geographical area, as shown in Fig. 1. Table I lists the notation in this paper. The primary network is assigned a certain spectrum band for its communication. Suppose scheduling is done in the time domain, with T time slots in a frame. For the primary network, it performs scheduling for transmission/reception without any consideration of the secondary network. A secondary node, however, is allowed to transmit in a time slot only if it is able to cancel its interference to its neighboring primary receivers. We assume the primary nodes are single antenna nodes. Suppose that there is a set of sessions $\tilde{\mathcal{F}}$ in the primary network $\mathcal{P}$. Each session has a source node and a destination node and traverses multi-hop relay nodes as needed. The route from a session's source node to its destination node is given a prior, which may be found by some standard routing protocols (e.g., AODV [12], DSR [9]). Denote $\tilde{\mathcal{L}}$ as a set of links in the network that are traversed by the active sessions in $\tilde{\mathcal{F}}$. Suppose the set of links $\tilde{\mathcal{L}}$ is operating under a feasible scheduling solution (for transmisson/reception) for the primary sessions $\tilde{\mathcal{F}}$, where interference at a primary receive node is avoided either through time slot or sufficient spatial separation. Since each primary node has only a single antenna, it can transmit at most one data stream to another node in a time slot.

For the secondary network, we assume each node is equipped with MIMO, which offers IC capability that is needed to achieve TC. We assume the number of antennas at a secondary node $i \in \mathcal{S}$ is A_i. For the secondary network $\mathcal{S}$, suppose that there is a set of sessions $\mathcal{F}$ in $\mathcal{S}$. Similar to a primary session, a secondary session has a source node, a destination node, and traverses multi-hop relay nodes as

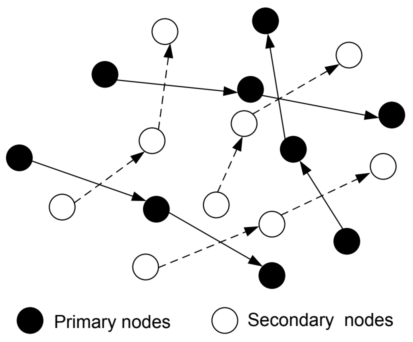


Fig. 1. A multi-hop secondary network co-located in the same area as a multi-hop primary network.

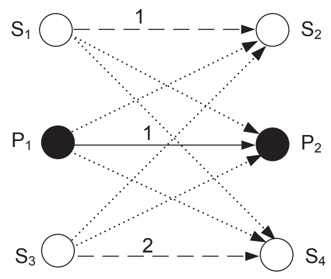


Fig. 2. A simple example illustrating SM and IC. A solid line represents the primary link, a dashed line represents a secondary link, and a dotted line represents an interference.

needed. The route from a secondary session's source node to its destination node is again given a priori. Denote $\mathcal{L}$ as the set of links that are traversed by any session in $\mathcal{F}$.

We use DoFs at a secondary node (no more than the number of antennas at the node) to represent its available resources. A DoF can be used for SM or IC. For SM, transmitting one data stream requires one DoF at the transmitter and one DoF at the receiver. In practice, the data rate carried in each data stream may vary with different channel conditions. For simplicity, we assume that the fixed modulation and coding scheme (MCS) is used for a link's data stream transmission, and one data stream in one time slot corresponds to one unit data rate. On the other hand, DoF consumption for IC depends on whether the IC is done at the transmitter or receiver. We use a simple example to illustrate this point. In Fig. 2, suppose P_1 and P_2 are a pair of primary transmit and receive nodes, while S_1 and S_2, S_3 and S_4 are two pairs of secondary transmit and receive nodes. Suppose that both the primary nodes P_1 and P_2 have one antenna, and the secondary nodes S_1, S_2, S_3, and S_4 are each equipped with 4 antennas (4 DoFs). P_1 is transmitting 1 data stream to P_2, S_1 is transmitting 1 data stream to S_2, and S_3 is transmitting 2 data stream to S_4. For the interference from S_1 to S_4, either transmitter S_1 or receiver S_4 can cancel this interference. If S_1 is to cancel this interference, then it will use 2 DoFs since S_4 is receiving 2 data streams; if S_4 is to cancel this interference, then it will use 1 DoF since S_1 is transmitting 1 data stream. Note the difference in DoF consumptions in IC by different nodes.

As described, to achieve TC, the secondary nodes have the sole responsibility to cancel interference to/from the primary nodes (i.e., inter-network interference) and interference within the secondary network (i.e., intra-network interference). In this example, for inter-network IC, secondary nodes S_2 and S_4 need to cancel the interference from primary transmit node P_1 with 1 DoF, respectively; the secondary transmit nodes S_1 and S_3 need to cancel their interference to primary receive node P_2 with 1 DoF, respectively. For intra-network IC, the interference from S_1 to S_4 needs to be cancelled, either by S_1 (with 2 DoF) or by S_4 (with 1 DoF) as discussed earlier; the interference from S_3 to S_2 needs to be cancelled, either by S_3 (with 1 DoF) or by S_2 (with 2 DoFs).

To successfully perform inter- and intra-network IC, it is crucial for the secondary nodes to have accurate channel state information (CSI). We propose one particular scheme for the secondary nodes to obtain CSI between themselves and their neighboring primary and secondary nodes while remaining transparent to the primary nodes. The main idea of the scheme is to have the primary and secondary nodes send out known pilot signal so that neighboring secondary nodes can estimate CSI. This is the practice for current cellular networks and we will employ such a mechanism for both the primary and secondary networks. Suppose the pilot sequences from the primary and secondary nodes are publicly available (as in cellular networks). After the primary and secondary nodes send out the pilot sequences, the neighboring secondary nodes can compare the known pilot sequences with the actual received sequences signal from the senders for channel estimation. Based on the reciprocity property of a wireless channel, the estimated CSI can also be used as CSIT (channel state information at transmitter side) when the secondary node becomes a transmit node. Therefore, each secondary node can obtain complete CSI between itself and a neighboring primary or secondary node. There are many other schemes that have been proposed to address this issue (see, e.g., [4], [13], [18], [19], [21], [27]). We omit their discussions here to conserve space. But the point here is that there exist schemes that we can use to obtain the necessary CSI for the secondary nodes to perform inter- and intra-network IC.

In our design of distributed algorithm for the secondary nodes to achieve TC, we consider a throughput maximization problem, with the objective of maximizing the minimum achievable session rate (in terms of data streams) among all secondary sessions. We choose this objective since it focuses on the worst case (minimum) achievable secondary session throughput, which ensures fairness across all secondary sessions.

III. A Distributed Algorithm

We propose a distributed scheduling algorithm to the throughput maximization problem while meeting all IC requirements for the secondary nodes. As described in our network setting, the set of sessions $\tilde{\mathcal{F}}$ in the primary network are transmitting under a given feasible scheduling solution. To have the secondary sessions operate in the same set of time

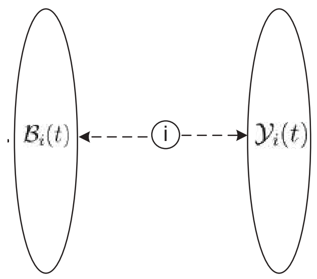


Fig. 3. Maintaining two local sets at node i to distinguish IC responsibility between node i and its neighboring nodes.

slots (to achieve TC), we employ MIMO at the secondary nodes for IC. The algorithm that we propose is an iterative greedy algorithm. We consider one link (from the set of links $\mathcal{L}$) at a time and try to increase the data streams on this link by 1 in this iteration. This increment is successful only if the transmitter, receiver and neighboring nodes of this link have enough remaining DoFs to cancel this new interference on neighboring primary and secondary nodes.

As discussed earlier, an interference can be canceled either by a secondary transmit or receive node. For efficient and feasible IC, a global node ordering scheme proposed in [14] would be useful. But such a global node ordering scheme is centralized in nature. Nevertheless, it gives us some hints in our design of distributed algorithm.

We propose to maintain two local sets at each node to keep track of the IC responsibility between this node and neighboring nodes. For example, at each secondary node $i \in \mathcal{S}$, we maintain one local set $\mathcal{B}_i(t)$ to store i's neighboring nodes that require node i to use its DoFs for IC and the other local set $\mathcal{Y}_i(t)$ to store i's neighboring nodes that use their own DoFs for canceling interference to/from node i (see Fig. 3). Note that there is no explicit node ordering among the nodes in sets $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$. By maintaining these two sets (with $\mathcal{B}_i(t)$ before node i and $\mathcal{Y}_i(t)$ after node i), we have achieved the desired efficiency in IC locally at node i. We will discuss the feasibility issue in Section IV.

The use of two local sets $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ at each secondary node i is centerpiece in our design of distributed scheduling algorithm to achieve TC. In our algorithm, we will exploit these two sets at each node to its fullest extent to achieve IC at the secondary nodes while meeting the resource constraints (limited DoFs at each node). In particular, when we find that a data stream cannot be further increased on a bottleneck link, we will consider moving some nodes from one local set into the other set so that the DoFs at a node can be re-allocated. This step is called adjusting IC responsibility in our algorithm (Step 3) and is a critical component to maximize the performance of our algorithm. At any iteration when this IC responsibility adjustment is not successful (and thus the number of data streams on the associated link cannot be further increased) for all time slots in a frame, our algorithm terminates.

TABLE II
State information at each node i

Symbol	Definition
$s_i(t)$	The status of node i in time slot t. $s_i(t)=$ Tx, Rx or Idle.
$\mathcal{B}_i(t)$	The set of nodes that node i allocates DoFs for IC to/from them in time slot t.
$\mathcal{Y}_i(t)$	The set of nodes that allocate their own DoFs for IC to/from node i in time slot t.
$\lambda_i^{\text{SM}}(t)$	The number of DoFs that node i has allocated for SM in time slot t.
$\lambda_i^{\text{IC}}(t)$	The number of DoFs that node i has allocated for IC in time slot t.
$\lambda_i^{\text{RM}}(t)$	The number of remaining DoFs at node $i \in \mathcal{S}$ in time slot t, i.e., $\lambda_i^{\text{RM}}(t) = A_i - \lambda_i^{\text{SM}}(t) - \lambda_i^{\text{IC}}(t)$.
$\tilde{\alpha}_i(t)$	The total number of data streams from node i's neighboring primary transmitters in time slot t.
$\tilde{\beta}_i(t)$	The total number of data streams received by node i's neighboring primary receivers in time slot t.
$\alpha_i(t)$	The total number of data streams from node i's neighboring secondary transmitters in time slot t.
$\beta_i(t)$	The total number of data stream received by node i's neighboring secondary receivers in time slot t.
$z_{i,j}(t)$	The number of data streams from transmit node i to receive node j.

A. State Information at Secondary Nodes

The state information that needs to be maintained at a secondary node (say i) is shown in Table II.

Local sets $\mathcal{B}_i(t)$ **and** $\mathcal{Y}_i(t)$: For each interference involving node i, it can be canceled by either node i or the other node involved in this interference. To explicitly distinguish who is responsible for IC for each interference, we maintain two local sets $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ at each node i, as shown in Figure 3. We denote $\mathcal{B}_i(t)$ as the set of secondary nodes that node i ($i \in \mathcal{S}$) allocates DoFs to cancel interference to/from them, and denote $\mathcal{Y}_i(t)$ as the set of secondary nodes that allocate their DoFs to cancel interference to/from i. At the beginning of our algorithm, we initialize $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ as empty sets, i.e., $\mathcal{B}_i(t) = \emptyset$ and $\mathcal{Y}_i(t) = \emptyset$ for $i \in \mathcal{S}$.

Accounting of DoF resource: In Table II, $z_{i,j}(t)$ represents the number of data stream transmitted from node i to node j. $\lambda_i^{\text{SM}}(t)$ and $\lambda_i^{\text{IC}}(t)$ represents the number of DoFs allocated for SM and IC at secondary node i in time slot t, respectively. $\lambda_i^{\text{RM}}(t)$ represents the number of remaining DoFs at a node i in time slot t. At the beginning of our algorithm, the status of each node $i \in \mathcal{S}$ is set to Idle, i.e., $s_i(t)$ =Idle for $t = 1, 2, \cdots, T$. Then, the initial DoF allocation for SM and IC at each node is 0. We have $\lambda_i^{\text{SM}}(t) = \lambda_i^{\text{IC}}(t) = 0$, $\lambda_i^{\text{RM}}(t) = A_i$ and $z_{i,j}(t) = 0$ for $i, j \in \mathcal{S}, t = 1, 2, \cdots, T$ in the initialization stage. $\tilde{\alpha}_i(t)$ and $\tilde{\beta}_i(t)$ are constants and are calculated based on active sessions in the primary network. These can be derived by the secondary nodes through monitoring/sensing of the neighboring primary nodes's activities. On the other hand, the initial values for $\alpha_i(t)$ and $\beta_i(t)$ are 0.

For these state information, except that $\tilde{\alpha}_i(t)$ and $\tilde{\beta}_i(t)$ are constants, the values for $s_i(t)$, $\mathcal{B}_i(t)$, $\mathcal{Y}_i(t), \lambda_i^{\text{SM}}(t)$, $\lambda_i^{\text{IC}}(t)$, $\lambda_i^{\text{RM}}(t)$, $z_{i,j}(t)$, $\alpha_i(t)$ and $\beta_i(t)$ are variables and will be updated during each iteration of the algorithm.

B. Step 1: Choosing a Link

To make a rate increment of each session by 1 DoF is equivalent to increasing the DoF on each active link by 1 DoF

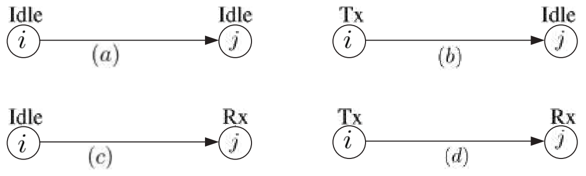


Fig. 4. Four cases of link status.

if each active link is traversed by 1 session. In the general case when an active link is traversed by multiple sessions, we need to increase the DoFs on this active link by multiple times, each for one session. In our distributed algorithm, we choose an active link for increment during an iteration. If a link is traversed by multiple sessions, then it is necessary to represent the link multiple times so that each session traversing this link is to be considered for data stream increment. Suppose there are k sessions traversing a link $l \in \mathcal{L}$. Then we represent link l by k *logical* links. We want to set a round robin for these logical links for rate increment so that each logical link is considered once in each cycle.

To do this, we employ the so-called distributed ranking algorithm by Zaks [28]. This algorithm was designed to solve the problem of sorting and ranking n processors in a distributed system. The input is an initial value unique for each processor. The output is a ranking of all n processors. To apply the distributed ranking algorithm, we assign an initial value for each logical link. Each initial value is generated randomly and guaranteed to be unique (under a reasonably good random number generator). We let the transmitter of each logical link to maintain the logical link's rank. After a logical link obtains its rank, it will know precisely when it will be considered for data stream increment.

C. Step 2: Data Stream Increment

After we identify a logical link (in Step 1), our algorithm will try to increase one data stream on the selected link, while satisfying IC constraints and transparency to the primary network.[1] We first present the necessary conditions under which one more data stream can be added on the link in a time slot. Then we describe how to update state information on the nodes that are involved in this increment.

Sufficient Conditions for Data Stream Increment. We now discuss when the number of data streams on a chosen link can be increased by 1 in a given time slot. Suppose link (i, j) is the link. Then both nodes i and j first check their current status ("Tx", "Rx", or "Idle"). Some cases can be clearly ruled out for consideration, i.e., $s_i(t) = \text{Rx}$ or $s_j(t) = \text{Tx}$. In these cases, link (i, j) cannot be considered for data stream increment in time slot t and we move to the next time slot $(t+1)$ immediately. When link (i, j) is suitable for data stream increment, there are four possible statuses as shown in Figure 4. The sufficient conditions for data stream increment on link (i, j) are as follows.

Case (a): $s_i(t) = \text{Idle}$ and $s_j(t) = \text{Idle}$.

[1]We drop the fine distinction between "link" and "logical link" when there is no confusion.

- $s_i(t) = \text{Idle}$: Since node i is idle, the local sets $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ are empty. We need to establish the sets $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ (see Figure 3) to decide the IC relationships between node i and its neighboring secondary receive nodes that will be interfered by i. We can put all these neighboring receive nodes in time slot t either in $\mathcal{B}_i(t)$ or $\mathcal{Y}_i(t)$.
  - If all neighboring receive nodes are put into $\mathcal{Y}_i(t)$, then the interference from node i to them will be canceled by these receive nodes. The following two conditions must be satisfied: (i) the total number of DoFs at node i should be greater than the total number of data streams received by its neighboring primary receivers, i.e., $A_i > \tilde{\beta}_i(t)$, (ii) all secondary receivers that are in $\mathcal{Y}_i(t)$ must have at least one remaining DoFs to cancel one more data stream interference from node i.
  - If all neighboring receive nodes are put into $\mathcal{B}_i(t)$, node i needs to cancel its interference to all these neighboring receive nodes. The following condition must be satisfied: the total number of DoFs at node i is more than the sum of data streams received by both neighboring primary and secondary receivers, i.e., $A_i > \tilde{\beta}_i(t) + \beta_i(t)$.
- $s_j(t) = \text{Idle}$: Similar to node i, we put node j's neighboring transmit nodes in either $\mathcal{B}_j(t)$ or $\mathcal{Y}_j(t)$. The sufficient conditions for j are similar as i, we omit its discussion here.

If the conditions for $s_i(t) = \text{Idle}$ and $s_j(t) = \text{Idle}$ are both satisfied, we proceed with this increment and update state information at nodes i, j and their neighboring nodes according to Figure 5 and Figure 6.

```
State update at idle node $i$ and neighboring receiver $k$
1. If $s_i(t) = \text{Idle}$:
2.    Update $s_i(t) = \text{Tx}$; $\lambda_i^{\text{SM}}(t) \leftarrow \lambda_i^{\text{SM}}(t) + 1$;
          $\lambda_i^{\text{RM}}(t) \leftarrow \lambda_i^{\text{RM}}(t) - 1$; $z_{i,j}(t) \leftarrow z_{i,j}(t) + 1$.
3.    If $\mathcal{Y}_i(t) \leftarrow$ {Neighboring active secondary receivers.}
4.       Update $\lambda_i^{\text{IC}}(t) \leftarrow \tilde{\beta}_i(t)$; $\lambda_i^{\text{RM}}(t) \leftarrow \lambda_i^{\text{RM}}(t) - \lambda_i^{\text{IC}}(t)$;
5.       For each receive node $k \in \mathcal{Y}_i(t)$:
6.          $\mathcal{B}_k(t) \leftarrow \mathcal{B}_k(t) \cup \{i\}$; $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) + 1$;
            $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) - 1$.
7.    Else if $\mathcal{B}_i(t) \leftarrow$ {Neighboring active secondary receivers.}
8.       Update $\lambda_i^{\text{IC}}(t) \leftarrow \tilde{\beta}_i(t) + \beta_i(t)$; $\lambda_i^{\text{RM}}(t) \leftarrow \lambda_i^{\text{RM}}(t) - \lambda_i^{\text{IC}}(t)$.
9.       For each node $k \in \mathcal{B}_i(t)$: $\mathcal{Y}_k(t) = \mathcal{Y}_k(t) \cup \{i\}$.
```

Fig. 5. Pseudocode to update state information when $s_i(t) = \text{Idle}$.

```
State update at idle node $j$ and neighboring transmitter $k$
1. If $s_j(t) = \text{Idle}$:
2.    Update $s_j(t) = \text{Rx}$; $\lambda_j^{\text{SM}}(t) \leftarrow \lambda_j^{\text{SM}}(t) + 1$;
          $\lambda_j^{\text{RM}}(t) \leftarrow \lambda_j^{\text{RM}}(t) - 1$; $z_{i,j}(t) \leftarrow z_{i,j}(t) + 1$.
3.    If $\mathcal{Y}_j(t) \leftarrow$ {Neighboring active secondary transmitters.}
4.       Update $\lambda_j^{\text{IC}}(t) \leftarrow \tilde{\alpha}_j(t)$; $\lambda_j^{\text{RM}}(t) \leftarrow \lambda_j^{\text{RM}}(t) - \lambda_j^{\text{IC}}(t)$
5.       For each transmit node $k \in \mathcal{Y}_j(t)$:
6.          $\mathcal{B}_k(t) \leftarrow \mathcal{B}_k(t) \cup \{j\}$; $\lambda_k^{\text{IC}}(t) = \lambda_k^{\text{IC}}(t) + 1$;
            $\lambda_k^{\text{RM}}(t) = \lambda_k^{\text{RM}}(t) - 1$.
7.    Else if $\mathcal{B}_j(t) \leftarrow$ {Neighboring active secondary transmitters.}
8.       Update $\lambda_j^{\text{IC}}(t) \leftarrow \tilde{\alpha}_j(t) + \alpha_j(t)$; $\lambda_j^{\text{RM}}(t) \leftarrow \lambda_j^{\text{RM}}(t) - \lambda_j^{\text{IC}}(t)$.
9.       For each $k \in \mathcal{B}_j(t)$: $\mathcal{Y}_k(t) \leftarrow \mathcal{Y}_k(t) \cup \{j\}$.
```

Fig. 6. Pseudocode to update state information when $s_j(t) = \text{Idle}$.

Case (b): $s_i(t) = \text{Tx}$ and $s_j(t) = \text{Idle}$.

- $s_i(t) = \text{Tx}$: In this case, the following conditions must be satisfied if node i wants to increase one more data stream on link (i, j): (i) node i has at least one remaining DoF for SM, i.e., $\lambda_i^{\text{RM}}(t) \geq 1$; (ii) each receive node $k \in \mathcal{Y}_i(t)$ has at least one remaining DoF to cancel the new interference from node i.
- $s_j(t) = \text{Idle}$: This case has been discussed in *Case (a)*.

If the conditions for $s_i(t) = \text{Tx}$ and $s_j(t) = \text{Idle}$ are both satisfied, we proceed with this increment and update state information at nodes i, j and their neighboring nodes according to Figure 6 and Figure 7.

```
State update at transmit node $i$ and neighboring receiver $k$
1. If $s_i(t) = \text{Tx}$:
2.   Update $\lambda_i^{\text{SM}}(t) \leftarrow \lambda_i^{\text{SM}}(t) + 1$; $\lambda_i^{\text{RM}}(t) \leftarrow \lambda_i^{\text{RM}}(t) - 1$;
         $z_{i,j}(t) = z_{i,j}(t) + 1$.
3.   For each receive node $k \in \mathcal{Y}_i(t)$:
4.     Update $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) + 1$; $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) - 1$.
```

Fig. 7. Pseudocode to update state information when $s_i(t) = \text{Tx}$.

Case (c): $s_i(t) = \text{Idle}$ and $s_j(t) = \text{Rx}$.

- $s_i(t) = \text{Idle}$: This case has been discussed in Case (a).
- $s_j(t) = \text{Rx}$: In this case, the following condition must be satisfied if node j wants to increase one more data stream on link (i, j): (i) node j has at least one remaining DoF for SM, i.e., $\lambda_j^{\text{RM}}(t) \geq 1$; (ii) each transmit node $k \in \mathcal{Y}_j(t)$ has at least one remaining DoF to cancel its interference to node j.

If the conditions for $s_i(t) = \text{Idle}$ and $s_j(t) = \text{Rx}$ are both satisfied, we proceed with this increment and update state information at nodes i, j and their neighboring nodes according to Figure 5 and Figure 8.

```
State update at receive node $j$ and neighboring transmitter $k$
1. If $s_j(t) = \text{Rx}$:
2.   Update $\lambda_j^{\text{SM}}(t) \leftarrow \lambda_j^{\text{SM}}(t) + 1$; $\lambda_j^{\text{RM}}(t) \leftarrow \lambda_j^{\text{RM}}(t) - 1$;
         $z_{i,j}(t) = z_{i,j}(t) + 1$.
3.   For each transmit node $k \in \mathcal{Y}_j(t)$:
4.     Update $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) + 1$; $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) - 1$.
```

Fig. 8. Pseudocode to update state information when $s_j(t) = \text{Rx}$.

Case (d): $s_i(t) = \text{Tx}$ and $s_j(t) = \text{Rx}$. The case for $s_i(t) = \text{Tx}$ has been discussed in *Case (b)* and $s_j(t) = \text{Rx}$ has been discussed in *Case (c)*. If the conditions for $s_i(t) = \text{Tx}$ and $s_j(t) = \text{Rx}$ are both satisfied, we proceed with this increment and update state information at nodes i, j and their neighboring nodes according to Figure 7 and Figure 8.

Recall that there are T time slots in a time frame. Node activities (both primary and secondary) and interference patterns in each time slot are different. If the data stream increment operation described above fails in the first time slot, we try it again in the second time slot and so forth, until a data stream increment is successful in a time slot or fails after all T time slots.

D. Step 3: Adjusting a Node's IC Responsibility

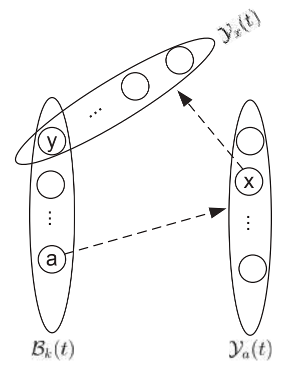
If the sufficient conditions at either node i or node j cannot be satisfied, we move on to this step. The only reason why link (i, j) fails to increase one data stream in step 2 is the lack of DoF resources at some nodes (bottleneck nodes), i.e., node i, j or nodes in $\mathcal{Y}_i(t)$ and $\mathcal{Y}_j(t)$. Since a node's local sets $\mathcal{B}$ and $\mathcal{Y}$ directly affects its DoF consumption for IC, we will try to swap some nodes between the sets $\mathcal{B}$ and $\mathcal{Y}$, and thus change their IC responsibilities. For example, if node k is short on DoFs, we can move some node $m \in \mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$, thereby transferring the IC responsibility from k to m. Through this change, some new DoF resources for the bottleneck node k become available, possibly allowing a new data stream increment to be made on the link under consideration.

The main idea of this step is as follows. For each time slot t, we identify the set of bottleneck nodes (denoted as $\mathcal{D}_{(i,j)}(t)$), which do not have enough remaining DoF resources should one more data stream is added onto link (i, j). For each node $k \in \mathcal{D}_{(i,j)}(t)$, we adjust node k's IC responsibility by moving some other nodes in $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$. To ensure feasibility, only a subset of nodes (denoted as $\bar{\mathcal{B}}_k(t)$), $\bar{\mathcal{B}}_k(t) \subseteq \mathcal{B}_k(t)$, is eligible for moving from $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$. After identifying $\bar{\mathcal{B}}_k(t)$ for k, we consider nodes in $\bar{\mathcal{B}}_k(t)$ in the order of non-increasing remaining DoFs, i.e., starting with the one that has the maximum remaining DoF (denoted as node a) if it is moved to $\mathcal{Y}_k(t)$. If this movement is infeasible, then our attempted adjustment fails in this time slot and we move on to the next time slot. Otherwise, we move a from $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$ and update their state information. After this movement, if a new data stream can be added on link (i, j), we are done. Otherwise, we continue moving the next node in $\bar{\mathcal{B}}_k(t)$ that has the maximum remaining DoF (denoted as node b) to $\mathcal{Y}_k(t)$ following the same process. This step terminates upon a new data stream can be successfully added on link (i, j) or all nodes in $\mathcal{D}_{(i,j)}(t)$ are considered for all time slots in a frame. In the rest of this section, we give more details for this idea.

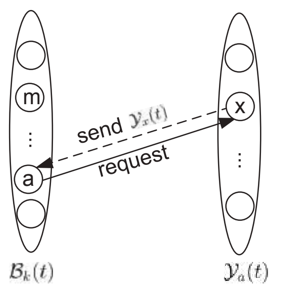
Finding bottleneck nodes $\mathcal{D}_{(i,j)}(t)$: $\mathcal{D}_{(i,j)}(t)$ can be easily found by identifying those nodes that would need more DoFs than their remaining DoFs should one more data stream were added on link (i, j).

Node sequence in $\mathcal{D}_{(i,j)}(t)$: To consider nodes one at a time in $\mathcal{D}_{(i,j)}(t)$ in a distributed environment, we could use a token to pass along from one node to the next so that at any time, only one node is considered for adjustment. There is no preference on which node to start but for the rest of the discussion, we assume that we start with node i, then j, before the other nodes in $\mathcal{D}_{(i,j)}(t)$. Note that a token is passed to the next node in $\mathcal{D}_{(i,j)}(t)$ only if the adjustment in the previous node is successful. Otherwise, the algorithm moves on to the next time slot in the frame.

Finding eligible subset nodes for swapping: Suppose the token is now passed onto node $k \in \mathcal{D}_{(i,j)}(t)$. To adjust node k's IC responsibility, we want to move one or more nodes in $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$, thus relieving node k's IC responsibility for these nodes. But for feasibility, not every node in $\mathcal{B}_k(t)$ is eligible for swapping. Now we discuss how to identify a subset of nodes $\bar{\mathcal{B}}_k(t)$ that are eligible to be moved to $\mathcal{Y}_k(t)$. By "eligible", we mean that when we move the subset of nodes from $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$, the IC responsibilities for all other nodes in $\bar{\mathcal{B}}_k(t)$ and $\mathcal{Y}_k(t)$ are not affected. We propose a sufficient condition to check whether or not a node is an eligible node as follows.



(a) An illustration of determining the eligibility of receive node $a \in \mathcal{B}_k(t)$.



(b) Steps involved in determining the eligibility of receive node $a \in \mathcal{B}_k(t)$.

Fig. 9. Determining the eligibility of receive node $a \in \mathcal{B}_k(t)$ when node k is a transmit node.

Suppose node k is a transmitter. Then it can consider those receive nodes in $\mathcal{B}_k(t)$ for moving to $\mathcal{Y}_k(t)$. We denote $c \leftarrow b$ as node b cancels interference to or from c. For a receive node $a \in \mathcal{B}_k(t)$, it can be moved to $\mathcal{Y}_k(t)$ if the following conditions are satisfied. For each transmit node $x \in \mathcal{Y}_a(t)$ that needs to do IC to a (i.e., $a \leftarrow x$), there cannot exist a receive node $y \in \mathcal{Y}_x(t)$ that y handles IC from x (i.e., $x \leftarrow y$), and k handles IC to y (i.e., $y \leftarrow k$) (see Figure 9 (a)). That is, there does not exist a receive node y, such that the following IC relationship holds: $a \leftarrow x \leftarrow y \leftarrow k$. If this condition is satisfied and a's remaining DoFs is at least one after moving to $\mathcal{Y}_k(t)$, a is an eligible node; otherwise, a is not. In Section IV, we will show that this condition can guarantee IC feasibility among all nodes.

To do this check, we have node a send a request for state information to those transmit nodes in $\mathcal{Y}_a(t)$. Upon receiving this request, each transmit node $x \in \mathcal{Y}_a(t)$ will send its state information $\mathcal{Y}_x(t)$ to node a (see Figure 9(b)). Upon receiving this state information, node a can check whether some receive nodes in $\mathcal{Y}_x(t)$ are also in $\mathcal{B}_k(t)$. If none of these receive nodes are in $\mathcal{B}_k(t)$ *and* a's remaining DoFs is at least one after moving to $\mathcal{Y}_k(t)$, then a is eligible. Otherwise, a is not eligible.

The above discussion is for the case when node k is a transmit node. The case when node k is a receive or idle node can be handled in a similar manner.

Moving node(s) in $\bar{\mathcal{B}}_k(t)$ to $\mathcal{Y}_k(t)$: Assume node $a \in \bar{\mathcal{B}}_k(t)$ has the maximum remaining DoFs after moving to $\mathcal{Y}_k(t)$. If $\bar{\mathcal{B}}_k(t) = \emptyset$, there is no eligible node and we move to next time slot immediately. Otherwise, at node k, we move a from $\bar{\mathcal{B}}_k(t)$ to $\mathcal{Y}_k(t)$ while at node a, we move k from $\mathcal{Y}_a(t)$ to $\mathcal{B}_a(t)$, and update their state information as follows.

- Case $s_k(t) = \text{Tx}$ or $s_k(t) = \text{Rx}$: In this case, k only needs to release one DoF. Since at node k, we move a from $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$ while at node a, we move k from $\mathcal{Y}_a(t)$ to $\mathcal{B}_a(t)$, then at least one DoF can be released from k. The node k updates the state information based on Figure 10, and the node a updates its state information based on Figure 11.
- Case $s_k(t) = \text{Idle}$: Recall that for the bottleneck node k in $\mathcal{D}_{(i,j)}(t)$, it might be i, j, or node in $\mathcal{Y}_i(t)$ or in $\mathcal{Y}_j(t)$. Since $\mathcal{Y}_i(t)$ represents the set of i's neighboring receive nodes that should allocate DoFs to cancel interference from node i, and $\mathcal{Y}_j(t)$ represents the set of j's neighboring transmit nodes that should allocate DoFs to cancel their interference to node j, then all nodes in $\mathcal{Y}_i(t)$ and $\mathcal{Y}_j(t)$ are active nodes. Therefore, node k can only represent node i or node j. Let's consider the case when node k is node i. The case when node k is node j is similar. Recall that when $s_k(t) = \text{Idle}$, both $\mathcal{B}_k(t)$ and $\mathcal{Y}_k(t)$ are empty. We establish $\mathcal{B}_k(t)$ and $\mathcal{Y}_k(t)$ by putting all neighboring active nodes in either $\mathcal{B}_k(t)$ or $\mathcal{Y}_k(t)$. Clearly, putting all neighboring receive nodes in $\mathcal{Y}_k(t)$ will add additional IC burden on all these nodes in $\mathcal{Y}_k(t)$ and may require adjusting each node's IC responsibility in $\mathcal{Y}_k(t)$. On the other hand, putting all neighboring receive nodes to $\mathcal{B}_k(t)$ will not have this issue as the IC responsibility on those nodes in $\mathcal{B}_k(t)$ are not affected and we only need to focus on adjusting node k's responsibility with one node in $\mathcal{B}_k(t)$. We adopt the latter approach and put all neighboring receive nodes in $\mathcal{B}_k(t)$ and set $\mathcal{Y}_k(t) = \emptyset$ (see Figure 12 (a)). For each node $p \in \mathcal{B}_k(t)$, node k is added to $\mathcal{Y}_p(t)$. Therefore, $\lambda_k^{\text{IC}}(t) = \sum_{n \in \mathcal{B}_k(t)}^{s_n(t)=\text{Rx}} \lambda_n^{\text{SM}}(t) + \tilde{\beta}_k(t)$, where $\tilde{\beta}_k(t)$ is the total number of data streams received by node i's neighboring primary receivers, and $\lambda_k^{\text{RM}}(t) = A_k - \lambda_k^{\text{IC}}(t)$. We start to put node a (the node in $\bar{\mathcal{B}}_k(t)$ that has the maximum remaining DoFs after movement) (see Figure 12 (b)) into $\mathcal{Y}_k(t)$. Both nodes k and a's state information is updated based on Figures 10 and 11, respectively. For the new sets, if a new data stream can be added on link (i, j), we are done. Otherwise, we continue to move another node $b \in \bar{\mathcal{B}}_k(t)$ that has the maximum remaining DoFs after movement following the same process (see Figure 12(c)). The process terminates if node k has at least one remaining DoF or $\bar{\mathcal{B}}_k(t) = \emptyset$. For the latter case, the adjustment fails and we move on to the next time slot.

IV. Feasibility

In our design of distributed algorithm, for each node k, we put its neighboring nodes in two sets: $\mathcal{B}_k(t)$ and $\mathcal{Y}_k(t)$. For the set of nodes in $\mathcal{B}_k(t)$, node k is responsible to cancel its interference to them if k is a transmit node or cancel the interference from them if k is a receive node. For the ease of understanding, we can consider the set of nodes in $\mathcal{B}_k(t)$ being positioned *before* node k while the set of nodes in $\mathcal{Y}_k(t)$ being positioned *after* node k. That is, there is a relative ordering among nodes in $\mathcal{B}_k(t)$, node k, and nodes in $\mathcal{Y}_k(t)$. Under this notion, node k, being positioned after the set of nodes in $\mathcal{B}_k(t)$, is responsible to cancel interference to/from nodes in $\mathcal{B}_k(t)$. Note that we did not make a finer distinction of the relative positions (or ordering) among the set of nodes in $\mathcal{B}_k(t)$ or $\mathcal{Y}_k(t)$.

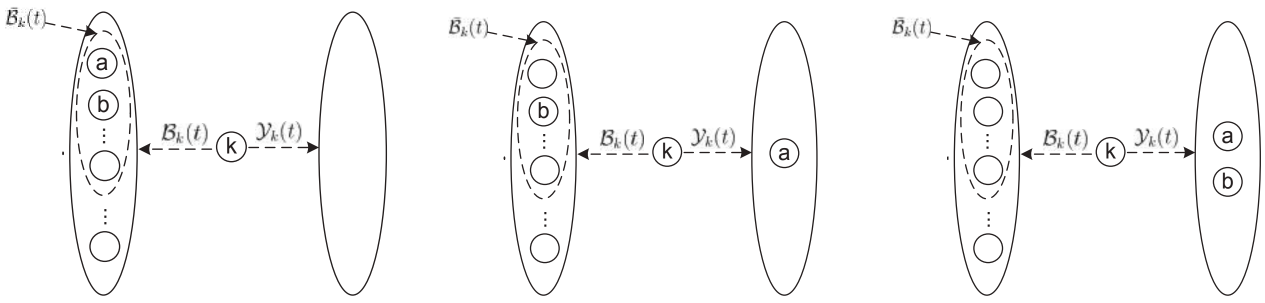
In this section, we show that the coarse set-based ordering $\mathcal{B}_k(t)$ or $\mathcal{Y}_k(t)$ at node k locally can in fact be mapped into a

```
**State Information Update at node $k$**
1. Update $\mathcal{B}_k(t) \leftarrow \mathcal{B}_k(t) - \{a\}$, $\mathcal{Y}_k(t) \leftarrow \mathcal{Y}_k(t) \cup \{a\}$;
2. If $s_k(t) = \text{Tx}$:
3.     Update $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) - (\lambda_a^{\text{SM}}(t) - z_{k,a}(t))$;
       $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) + (\lambda_a^{\text{SM}}(t) - z_{k,a}(t))$;
4. Else if $s_k(t) = \text{Rx}$:
5.     Update $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) - (\lambda_a^{\text{SM}}(t) - z_{a,k}(t))$;
       $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) + (\lambda_a^{\text{SM}}(t) - z_{a,k}(t))$;
6. Else if $s_k(t) = \text{Idle}$:
       $\lambda_k^{\text{IC}}(t) \leftarrow \lambda_k^{\text{IC}}(t) - \lambda_a^{\text{SM}}(t)$; $\lambda_k^{\text{RM}}(t) \leftarrow \lambda_k^{\text{RM}}(t) + \lambda_a^{\text{SM}}(t)$;
```

Fig. 10. Update state information at k.

```
**State Information Update at node $a$**
1. Update $\mathcal{Y}_a(t) \leftarrow \mathcal{Y}_a(t) - \{k\}, \mathcal{B}_a(t) \leftarrow \mathcal{B}_a(t) \cup \{k\}$;
2. If $s_k(t) = \text{Tx}$:
3.     Update $\lambda_a^{\text{IC}}(t) \leftarrow \lambda_a^{\text{IC}}(t) + (\lambda_k^{\text{SM}}(t) - z_{k,a}(t))$;
$\lambda_a^{\text{RM}}(t) \leftarrow \lambda_a^{\text{RM}}(t) - (\lambda_k^{\text{SM}}(t) - z_{k,a}(t))$;
4. Else if $s_k(t) = \text{Rx}$:
5.     Update $\lambda_a^{\text{IC}}(t) \leftarrow \lambda_a^{\text{IC}}(t) + (\lambda_k^{\text{SM}}(t) - z_{a,k}(t))$;
$\lambda_a^{\text{RM}}(t) \leftarrow \lambda_a^{\text{RM}}(t) - (\lambda_k^{\text{SM}}(t) - z_{a,k}(t))$;
6. Else if $s_k(t) = \text{Idle}$: No Changes;
```

Fig. 11. Update state information at a.



(a) Before movement (b) Move a to $\mathcal{Y}_k(t)$ (c) Move b to $\mathcal{Y}_k(t)$

Fig. 12. An illustration of movement process when node k is an idle node.

"global node ordering" for IC among all the nodes explicitly. More formally, we give the following definition.

Definition 1: A global node ordering for IC is a list of nodes where the position of a node in the list determines its IC responsibility. Based on this list, a node is responsible for canceling interference to/from these nodes that are before itself in the list; a node does not need to cancel the interference to/from those nodes that are after itself in the list as that interference will be canceled by those nodes.

Based on this definition, we show if there exists a global node ordering for IC among the active nodes in the network, then there exists a set of feasible precoding vectors at each secondary transmitter and decoding vectors at each secondary receiver so that all data (in both primary and secondary networks) can be transported free of interference using zero-forcing technique on the secondary nodes. That is, if a global node ordering for IC exists among the active nodes, then there exist feasible precoding and decoding vectors at the PHY layer to implement the desired IC and SM in the network.

Lemma 1: Upon the termination of the distributed algorithm, there exists a global node ordering for IC among all nodes in each time slot t.

Proof: Before we start our algorithm, all secondary nodes are inactive and there does not exist any ordering among the nodes. Since none of the primary nodes perform IC (due to the fact that potential interference among the primary nodes is handled by interference avoidance through time slot), we can envision a list containing all active primary nodes with arbitrary order among them. We will build upon this list to establish a global node ordering for IC.

To achieve TC, all interferences to/from primary network is canceled by the secondary nodes. Following the definition of global node ordering for IC, the secondary nodes must be placed after the primary nodes. We now show that we can maintain a global node ordering for IC at each iteration. Upon termination of the last iteration, the list remains a global node ordering for IC.

If a link fails to increase one data stream at the end of any iteration, the current global ordering for IC will not be affected. So in our proof, we only need to discuss the case that we can increase one data stream upon the end of the an iteration. Our proof is based on induction. For the first iteration, a secondary link is selected for data stream increment. We append the secondary transmit and receive nodes of this link at the end of the current list. Since there is no IC relationship between the secondary transmit and receive nodes of the chosen link, we can put transmit node either before or after the receive node. The new global ordering list consists of all active primary nodes, plus the secondary transmit and receive nodes of the chosen link. Since we can increase one data stream on this link, all interference from this link's transmit node to the neighboring primary receivers can be canceled by this transmit node, and all interference from neighboring primary transmitters to the chosen link's receive node can be canceled by this receive node. Obviously, this new list satisfies global node ordering for IC by definition.

Upon the end of n-th iteration, suppose there exists a global node ordering for IC. Then, we show that at the end of the $(n+1)$-th iteration, there still exists a global node ordering for IC. Denote link (i, j) as the link chosen for data stream increment during the $(n+1)$ iteration. We consider two cases: (i) a data stream can be added onto (i, j) without adjusting node ordering; (ii) a data stream can be added onto (i, j) but requiring adjusting node ordering:

- (i) We first consider that one data stream can be added onto (i, j) without adjusting node ordering in the current global node ordering list. We take node i as an example. There are two cases:

  - Node i is not yet on the current global node ordering list. In this case, $s_i(t)$ =Idle and our algorithm will put node i's neighboring receive nodes (not including j) either in $\mathcal{B}_i(t)$ or $\mathcal{Y}_i(t)$. Since all these neighboring receive nodes are active, they already have their positions in the current global ordering list in a previous iteration. If node i's neighboring receive nodes are put in $\mathcal{B}_i(t)$, it is the same as putting node i after these neighboring receive nodes. If node i's neighboring receive nodes are put in $\mathcal{Y}_i(t)$, it is the same as putting node i before these neighboring receive nodes. In either case, other nodes' relative ordering on the current global node ordering list is not affected, and thus IC responsibilities among them remain the same. Since one data stream can be added onto link (i, j) successfully, node i must be able to cancel interference to these nodes in $\mathcal{B}_i(t)$ (if these receive nodes are put into $\mathcal{B}_i(t)$), or the interference from i can be canceled by these nodes in $\mathcal{Y}_i(t)$ (if these receive nodes are put into $\mathcal{Y}_i(t)$). Therefore, on the new list, each node is responsible for canceling interference to/from these nodes that are before itself in the list; each node does not need to cancel the interference to/from those nodes that are after itself in the list as that interference will be canceled by those nodes. By definition, the new list satisfies the global node ordering for IC.
  - Node i is already on the global node ordering list. In this case, $s_i(t)$=Tx and the node i only performs data stream increment. There is no new node to be added to the list or none of the nodes change its position in the list. Therefore, the current ordering list is the same as that in the n-th iteration and satisfies the global node ordering for IC.

The case for node j is similar and we omit its discussion here to conserve space.

- (ii) We then consider that one data stream can be added onto (i, j) but requiring adjusting node ordering in the current global node ordering. We assume that node $k \in \mathcal{D}_{(i,j)}(t)$ is under consideration for adjustment. Suppose k is a transmit node. Recall that a necessary condition for a receive node $a \in \mathcal{B}_k(t)$ to be moved to $\mathcal{Y}_k(t)$ is that: for each transmit node $x \in \mathcal{Y}_a(t)$ that needs to do IC to a (i.e., $a \leftarrow x$), there cannot exist a receive node $y \in \mathcal{Y}_x(t)$ that y handles IC from x (i.e., $x \leftarrow y$), and k handles IC to y (i.e., $y \leftarrow k$) (see Figure 9(a)). That is, there does not exist a transmit node x and a receive node y, such that the following IC relationship holds: $a \leftarrow x \leftarrow y \leftarrow k$. Therefore, when node a is chosen to move from $\mathcal{B}_k(t)$ to $\mathcal{Y}_k(t)$, node a's IC responsibility for other transmit nodes, and node k's IC responsibility for other receive nodes will not change, except changing the position of k and a (i.e., moving a after k). Since this change is successful, a is able to cancel the interference from k and other transmit nodes that are before k. Therefore, the new list satisfies the global node ordering for IC. The discussion when k is a receiver is similar.

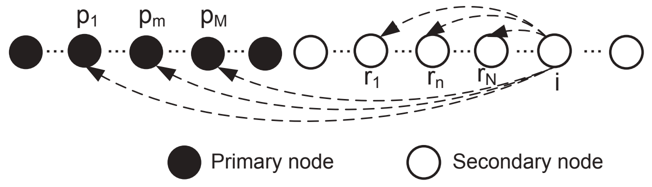


Fig. 13. A secondary transmit node i performs IC to neighboring primary and secondary receive nodes in a time slot t.

Therefore, we conclude that after the $(n + 1)$-th iteration, the new list satisfies the global node ordering for IC.

Based on the above discussion, we conclude that upon the termination of the distributed algorithm, we have a global node ordering for IC. ■

Theorem 1: There exists a set of feasible precoding vectors at each secondary transmitter and a feasible set of decoding vectors at each secondary receiver so that all data (in both primary and secondary networks) can be transported free of interference based on the global node ordering for IC.

Proof: We first consider a secondary transmit node i on the global node ordering list, as shown in Figure 13. The dashed arrows represent the interference from node i to those receive nodes that are before node i on the global node ordering list. The nodes $p_1 \cdots p_M$ are i's neighboring primary receivers, while nodes $r_1 \cdots r_N$ are i's neighboring secondary receivers. Suppose that i transmits $z_{(i,j)}$ data streams to secondary node j. Denote $\mathbf{u}_i^q$ as an $A_i \times 1$ transmit weight vector at i for each data stream q $(1 \leq q \leq z_{(i,j)})$, and $\mathbf{v}_j^q$ as an $A_j \times 1$ receive weight vector at receiver node j to receive data stream q. Since each primary link only transmits one data stream, we use $\mathbf{u}_p^1$ and $\mathbf{v}_p^1$ to denote the primary node p's $(p \in \{p_1, \cdots, p_M\})$ transmit and receive vectors .

Denote $\mathbf{H}_{(i,b)}$ $(b \in \{p_1, \cdots, p_M, r_1, \cdots, r_N\})$ as the $A_i \times A_b$ channel gain matrix between nodes i and b. We assume a rich scattering environment, where all channels $H_{i,b}$ have full rank and independent with each other. To successfully transmit $z_{(i,j)}$ data stream from node i to its intended receive node j, the transmit node i should cancel all its interference to primary receive nodes p_1 to p_M and secondary receive nodes r_1 to r_N. Then, we should have the following constraints:

$$(\mathbf{u}_i^q)^T \mathbf{H}_{(i,j)} \mathbf{v}_j^q = 1 \,, \qquad (1 \leq q \leq z_{(i,j)}) \,, \tag{1}$$

$$(\mathbf{u}_i^q)^T \mathbf{H}_{(i,j)} \mathbf{v}_j^d = 0 \,, \qquad (1 \leq q, d \leq z_{(i,j)}, d \neq q) \,, \tag{2}$$

$$(\mathbf{u}_i^q)^T \mathbf{H}_{(i,p_m)} \mathbf{v}_{p_m}^1 = 0 \,, \qquad (1 \leq q \leq z_{(i,j)}, 1 \leq m \leq M) \,, \tag{3}$$

$$(\mathbf{u}_i^q)^T \mathbf{H}_{(i,r_n)} \mathbf{v}_{r_n}^d = 0 \,, \qquad (1 \leq q \leq z_{(i,j)}, 1 \leq n \leq N, \; 1 \leq d \leq z_{(s_n,r_n)}) \,, \tag{4}$$

where s_n is the transmit node which transports $z_{(s_n,r_n)}$ data streams to secondary receive node r_n.

The number of constraints from (1) and (2) are $(z_{(i,j)})^2$. The number of constraints from (3) is $z_{(i,j)} \sum_{m=1}^{M} 1$. The number of constraints from (4) is $z_{(i,j)} \sum_{n=1}^{N} z_{(s_n,r_n)}$. The total number of constraints is therefore $(z_{(i,j)} \sum_{m=1}^{M} 1 + (z_{(i,j)})^2 + z_{(i,j)} \sum_{n=1}^{N} z_{(s_n,r_n)})$. Recall that in our algorithm, in either step 2 or step 3, the total number of DoF consumption cannot be more than the total number of DoFs at a node. We have, $z_{(i,j)} \sum_{m=1}^{M} 1 + (z_{(i,j)})^2 + z_{(i,j)} \sum_{n=1}^{N} z_{(s_n,r_n)} \leq z_{(i,j)}(\sum_{m=1}^{M} 1 + z_{(i,j)} + \sum_{n=1}^{N} z_{(s_n,r_n)}) \leq z_{(i,j)} A_i$. That is, the total number of constraints is no more than $z_{(i,j)} A_i$.

Since the precoding vector $\mathbf{u}_i^q$ is an $A_i \times 1$ vector for each data stream q ($1 \leq q \leq z_{(i,j)}$), the total number of variables at the transmit node i is $z_{(i,j)} A_i$ and the number of variables is no less than the number of constraints. On the other hand, since the channels $H_{(i,b)}$ are full rank and independent with each other, it can be shown that the constraints in (1), (2), (3), and (4) are linearly independent with each other based on [14]. So for any given $\mathbf{v}_j^q$ ($1 \leq q \leq z_{(i,j)}$), we are guaranteed to construct feasible precoding vectors $\mathbf{u}_i^q$ ($1 \leq q \leq z_{(i,j)}$) at transmit node i.

The proof that we can construct the feasible decoding vectors $\mathbf{v}_j^q$ ($1 \leq q \leq z_{(i,j)}$) at the secondary receive node j (for any given precoding vectors $\mathbf{u}_i^q$ ($1 \leq q \leq z_{(i,j)}$)) is similar to the transmit node i. We omit its discussion here to conserve space. Based on the above discussions, there exist feasible precoding/decoding vectors at the secondary nodes. Therefore, there exists a set of feasible precoding vectors at each secondary transmitter and a feasible set of decoding vectors at each secondary receiver so that all data (in both primary and secondary networks) can be transported free of interference. This completes the proof. ■

V. Complexity and Overhead Analysis

A. Complexity Analysis

We now show that the overall computation complexity of the distributed algorithm is polynomial time. Step 1 (ranking of active secondary links) is done only once. As shown in [28], this step can be done in $O(S^2)$. The iteration of our algorithm involves steps 2 and 3. We now analyze the complexity of each iteration and the number of iterations required in the algorithm.

In step 2, nodes i and j (for a chosen link (i,j)) need to check the feasibility of increasing one more data stream over at most T time slots. The worst case scenario is that both nodes i and j are idle (case (a) in Fig. 4). Since both nodes i and j need to check the number of remaining DoFs of each of their neighboring nodes and the number of DoFs used for SM by these nodes, the complexity of this operation is $O(2S)$. Afterward, nodes i and j, and their neighbors, need to update their DoF allocation status. The complexity of this operation is $O(S)$. Since there is a total of T time slots, the total complexity of this step is $T \cdot O(2S + S) = O(ST)$.

In step 3, nodes i and j, as well as their neighboring nodes attempt to adjust IC responsibility in at most T time slots. During each time slot, the computation consists of three parts: (i) identifying the subset of nodes $\mathcal{D}_{(i,j)}(t)$, which has a complexity $O(S)$; (ii) identifying the set of nodes $\bar{\mathcal{B}}_k(t)$ for each $k \in \mathcal{D}_{(i,j)}(t)$, which has a complexity of $O(S^2)$. Since the number of nodes in $\mathcal{D}_{(i,j)}(t)$ is at most S, the total complexity for (ii) is $O(S * S^2) = O(S^3)$; (iii) adjusting the IC responsibility for each node $k \in \mathcal{D}_{(i,j)}(t)$, and updating each node's state information, which has a complexity $O(S)$. Since there is a total of T time slots, the total complexity of step 3 is $T \cdot O(S + S^3 + S) = O(TS^3)$.

Since each node has A antennas and there are L active links in the network, the number of iterations of our algorithm is at most $O(LA)$. Therefore the overall complexity is $O(S^2 + O(LA) \cdot [O(ST) + O(TS^3)]) = O(LATS^3)$.

B. Overhead Analysis

In a distributed environment, the success of this distributed algorithm relies on the message exchanges in the common control channel among the secondary nodes. Since it is hard to quantify overhead precisely induced by this algorithm, we develop an upper bound for the total volume of message exchanges among the secondary nodes in this algorithm. In what follows, we analyze the volume of message exchanges in each step. In Step 1, message exchange is required to sort the links in the network. Here, we employ the distributed ranking algorithm based on the results in [28], which require $O(S^2)$ message exchanges in the worst case.

As for Step 2 and Step 3, there are at most $O(TLA)$ iterations. To find the volume of message exchange in each iteration, we analyze each step separately. In Step 2, both nodes i and j need to communicate with their neighboring nodes to check the remaining number of DoFs and the number of DoFs for SM by these nodes. This requires $O(S)$ message exchanges in each time slot. Afterward, nodes i and j need to send messages to their neighboring nodes to update their DoF allocation status. This requires $O(S)$ messages. So the total volume of message exchange in Step 2 is $O(S)$. In Step 3, for each bottleneck node, the volume of message exchange with a node in $\mathcal{B}_k(t)$ is $O(2)$ and there are at most $O(S)$ nodes in $\mathcal{B}_k(t)$. So Step 3 requires O(S) message exchanges for each bottleneck node. Since there are at most S bottleneck nodes, the total volume of message exchanges induced by Step 3 is $O(S^2)$. Putting things together, the total volume of message exchange induced by Step 2 and Step 3 is $O(TLA(S + S^2)) = O(TLAS^2)$.

By adding the message exchanges in the three steps together, this distributed algorithm requires $O(S^2 + TLAS^2)$ message exchanges in the worse case.

VI. Simulation Results

In this section, we present simulation results to demonstrate the performance of the proposed distributed algorithm. We compare our results with the centralized methods as discussed in [21]. Since the centralized problem formulation is MILP, which is NP-hard in general, we cannot obtain the optimal solution for comparison. Instead, we will compare the performance of our algorithm to an upper bound of the objective for the centralized problem. Such an upper bound can be obtained by running CPLEX for a given termination time. Clearly, such a comparison approach is very aggressive and conservation. This is because the optimal objective value (not obtainable) to the centralized problem lies between the upper bound and the feasible solution obtained by our distributed algorithm. Therefore, if the feasible solution from our distributed algorithm is somehow close to the upper bound by CPLEX, then we can claim that our solution (objective) is even closer to the optimal objective and thus is competitive.

A. Simulation Setting

We consider a secondary CR network co-locates with a primary network within a 100×100 area. For generality, we normalize the units for distance, bandwidth, and data rate with appropriate dimensions. Each node (both primary and secondary) is randomly deployed inside the 100×100 area. The primary nodes are traditional single-antenna node while the secondary nodes are equipped with MIMO, with four antennas on each node. We assume that each node's transmission range and interference range are 30 and 50, respectively. We assume a time frame is divided into $T = 10$ time slots.

B. A Case Study

Before we present complete results, we show results for one network instance, with 20 primary nodes and 30 secondary nodes. The location of each node is shwon in Figure 14. We assume there are three primary sessions and four secondary sessions, with each session's source and destination nodes shown in Figure 14. For simplicity, we assume that minimum-hop routing is used for each primary and secondary sessions, although other routing methods may be used if needed. Figure 14 shows the routing topology for each primary and secondary sessions, where a solid line represents a primary link and a dashed line represents a secondary link. Scheduling for the primary and secondary links is given in this figure, where numbers in the box represent the time slots used by the corresponding link. Note that scheduling for the primary links is solely determined by the primary network, while scheduling for each secondary link is found by our distributed algorithm.

The objective value obtained from our distributed algorithm is 0.6 (in less than a second computational time). On the other hand, the upper bound obtained by CPLEX is 0.7 (with a cut-off time of 8 hours). The CPLEX solver is run on a Dell Precision T7600 workstation, with dual Intel Xeon CPUE5-2687W CPUs (each with 8 cores) running at 3.1 GHz. The memory of the workstation is 64 GB and the OS is Windows 7 Professional. As discussed, since the optimal solution lies between 0.6 and 0.7, our objective value (0.6) is quite close to the unknown optimal.

To show whether TC is achieved by the secondary network, we consider one time slot, say 6. Figure 15 shows the set of active links in time slot 6 for both networks. In this time slot, secondary links $S_{28} \to S_{17}, S_{13} \to S_{24}, S_{30} \to S_{12}, S_3 \to S_1, S_4 \to S_{11}$ and $S_4 \to S_5$ are active simultaneous with primary links $P_1 \to P_8$ and $P_4 \to P_9$, through IC by the secondary nodes.

We first consider inter-network IC:

- For secondary link $S_{28} \to S_{17}$, its interference to P_9 on primary link $P_4 \to P_9$ is canceled by S_{28} with 1 DoF, while the interference from P_4 and P_1 to S_{17} is canceled by S_{17}, each with 1 DoF.
- For secondary links $S_3 \to S_1, S_{30} \to S_{12}, S_{13} \to S_{24}, S_4 \to S_{11}$ and $S_4 \to S_5$, the interference from their transmitters $(S_3, S_{30}, S_{13}, S_4)$ to receiver P_8 on primary link $P_1 \to P_8$ is canceled by S_3, S_{30}, S_{13} and S_4, each with 1 DoF. The interference from P_1 to S_{12} and S_{24} is canceled by S_{12} and S_{24} with 1 DoF, respectively, and the interference from P_4 to S_{11} is canceled by S_{11} with 1 DoF.

For intra-network IC within the secondary network, our solution shows that:

- S_{11} is canceling interference from S_3 and S_4, each with 1 DoF.
- S_5 is canceling interference from S_3 and S_4, each with 1 DoF.
- The interference from S_4 to S_1 is canceled by S_1 with 1 DoF.
- The interference from S_3 to S_{12} is canceled by S_{12} with 1 DoF.
- The interference from S_{13} to S_{12} is canceled by S_{13} with 2 DoFs.
- The interference from S_{30} to S_1 and S_{11} is canceled by S_{30}, each with 1 DoF.

The details of DoF allocation for SM and IC at each active secondary node in time slot 6 are shown in Table III. In this table, the second and third columns represent the set of secondary nodes that are in $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ (i.e., before and after this node in the global node ordering) in our distributed algorithm, respectively. The fourth column represents the number of DoFs allocated for SM. The fifth column represents the number of DoFs that are allocated for IC to/from primary network. The last column represents the number of DoFs allocated for IC for the set of secondary nodes in $\mathcal{B}_i(t)$.

TABLE III
DoF ALLOCATION FOR SM AND IC AT EACH ACTIVE SECONDARY NODE IN TIME SLOT 6.

Node i	$\mathcal{B}_i(t)$	$\mathcal{Y}_i(t)$	DoF for SM	IC to/from primary	DoF for IC within secondary network
S_1	$\{S_4\}$	$\{S_{30}\}$	1	0	2
S_3		$\{S_5, S_{11}, S_{12}\}$	1	1	0
S_4		$\{S_1, S_5, S_{11}\}$	2	1	0
S_5	$\{S_3, S_4\}$		1	0	2
S_{11}	$\{S_3, S_4\}$	$\{S_{30}\}$	1	1	2
S_{12}	$\{S_3\}$	$\{S_{13}\}$	1	1	1
S_{13}	$\{S_{12}\}$		1	1	2
S_{17}			1	2	0
S_{24}			1	1	0
S_{28}			1	1	0
S_{30}	$\{S_1, S_{11}\}$		1	1	2

Now, we show that there exists a global node ordering for IC among all nodes in time slot 6. Based on Table III, we can establish a global node ordering for IC among all nodes explicitly. Since none of the primary nodes perform IC, we put active primary nodes p_1, p_4, p_8 and p_9 in the front of global node ordering list with arbitrary order among them. Based on $\mathcal{B}_i(t)$ and $\mathcal{Y}_i(t)$ in Table III, we can establish a global ordering among the secondary nodes, as shown in Figure 16. The arrows originating from a node in the figure represent the interference from that node.

In this figure, we first take a receive node S_{12} as an example. S_{12} is being interfered by transmit nodes P_1, S_3 and S_{13}. Since

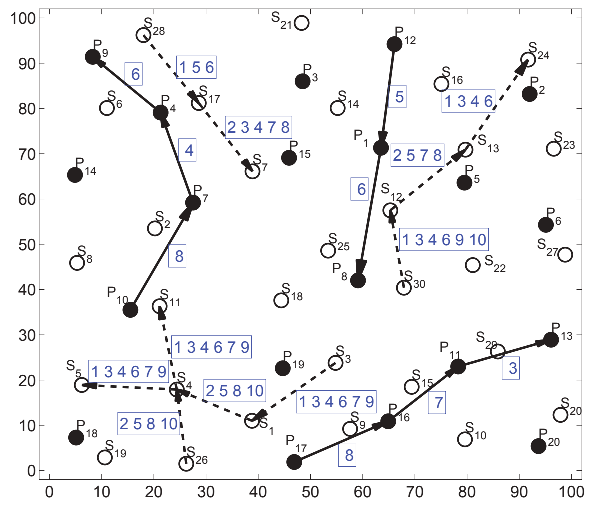


Fig. 14. Routing topology for each primary and secondary sessions and scheduling on each link of the respective route. The numbers in the box next to a link show the time slots when the link is active.

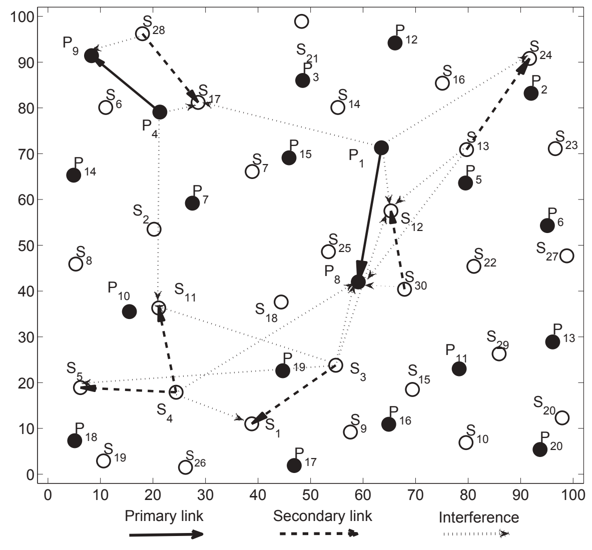


Fig. 15. Active links in time slot 6 in both primary and secondary networks.

P_1 and S_3 are before S_{12}, S_{12} is responsible for canceling their interference, each with 1 DoF. For the interference from S_{13}, S_{12} does not need to use any DoF to cancel this interference, since S_{13} is after S_{12} in this global node ordering. This interference is to be canceled by S_{13} with 2 DoFs. As a second example, consider transmit node S_3. S_3 is interfering receive nodes P_8, S_5, S_{11} and S_{12}. Since P_8 is before S_3, S_3 is responsible for canceling this interference with 1 DoF. For its interference to S_5, S_{11} and S_{12}, S_3 does not need to use any DoF to cancel this interference, since S_5, S_{11} and S_{12} are after S_3 in this global node ordering. This interference is canceled by S_5, S_{11} and S_{12}, respectively, each with 1 DoF. It is easy to verify that based on this global node ordering, the IC responsibilities at nodes S_4, S_{17}, S_{24}, S_{28}, S_5, S_{11}, S_1, S_{30} and S_{13} are all satisfied.

C. Comparison to Interweave Paradigm

To show the benefits of TC paradigm, we compare our results to those under the interweave paradigm. For the latter, a secondary node is not allowed to transmit (receive) at the same time when a nearby primary node is active. That is, the secondary nodes will not perform inter-network IC for interference to/from the primary nodes. The problem formulation for this paradigm is given in [22], which is similar to the problem formulation for TC paradigm except that we remove DoF allocation by the secondary nodes to cancel interference to/from the primary nodes. The problem formulation remains an MILP, and an upper bound can be obtained by running CPLEX for a given termination time (i.e., 8 hours).

Following the same setting as in the case study in the last section, we obtain an upper bound of 0.4 for the objective value (comparing to 0.6 from our distributed solution in Section VI-B). The time slot scheduling on each link of the secondary sessions is shown in Fig. 17. Comparing Fig. 14 and 17, we find that the set of time slots used by each secondary link under interweave paradigm is smaller. We take the link

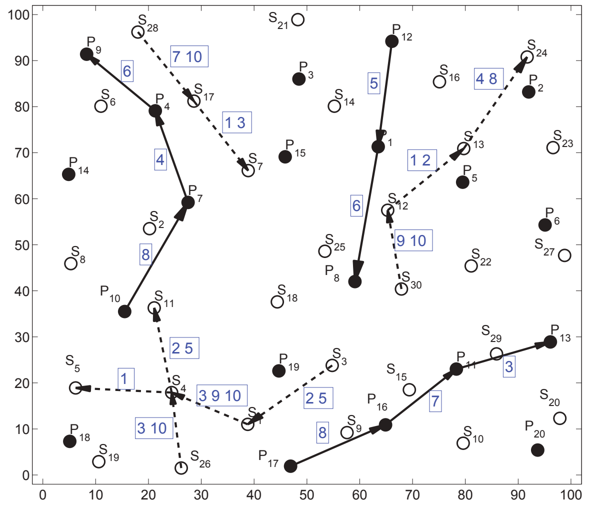


Fig. 17. Routing for each session and scheduling on each link for both primary and secondary networks under the interweave paradigm.

$S_{28} \rightarrow S_{17}$ as an example. Under interweave paradigm, this link cannot use time slot 6 as the neighboring primary link $P_4 \rightarrow P_9$ is using this it. However, under TC paradigm, this link can use time slot 6 to achieve the simultaneous activation with the primary link $P_4 \rightarrow P_9$. For any secondary link in Figure 17, we cannot find one that simultaneously actives with the primary links. There is no inter-network interference in the network.

D. Complete Results

We run our distributed algorithm for 50 random network instances, with 20-node primary network and 30-node secondary network. The number of primary and secondary sessions are random, with the source and destination nodes of each session are randomly generated. Table IV compares the objective values from our distributed algorithm and the upper bounds

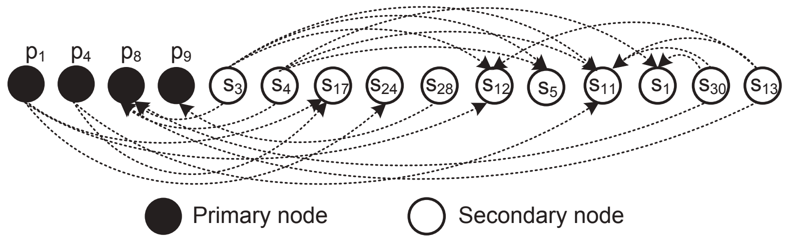


Fig. 16. A global node ordering for IC in time slot 6.

TABLE IV
RESULTS FOR 50 NETWORK INSTANCES.

Instance	Our Algorithm	CPLEX	Instance	Our Algorithm	CPLEX
1	0.8	0.9	26	0.7	0.8
2	0.7	0.9	27	0.6	0.7
3	0.4	0.5	28	0.5	0.7
4	0.4	0.4	29	0.5	0.6
5	0.4	0.6	30	0.7	0.8
6	0.5	0.6	31	1.0	1.1
7	0.9	1.1	32	0.8	1.0
8	0.7	0.8	33	0.3	0.4
9	1.1	1.1	34	0.7	0.9
10	0.3	0.3	35	0.5	0.6
11	0.6	0.7	36	0.8	0.9
12	0.7	0.8	37	0.6	0.8
13	0.3	0.4	38	0.5	0.5
14	0.9	1.0	39	0.6	0.7
15	0.7	0.8	40	0.4	0.4
16	1.0	1.0	41	0.6	0.7
17	0.9	1.0	42	0.8	0.9
18	0.2	0.4	43	0.3	0.3
19	0.6	0.6	44	0.5	0.7
20	0.6	0.7	45	0.4	0.5
21	1.1	1.1	46	0.6	0.6
22	0.6	0.7	47	0.8	0.9
23	0.8	0.8	48	0.4	0.5
24	0.6	0.9	49	0.5	0.6
25	0.6	0.6	50	0.8	1.0

from CPLEX solver. The average ratio between the two over 50 instances is 83.7%, with standard derivation of 0.073. Since the optimal objective value (unknown) to the centralized problem lies between the upper bound and the feasible solution obtained by our distributed algorithm, these results affirm that our distributed algorithm is highly competitive.

VII. CONCLUSIONS

TC is a new spectrum sharing paradigm that allows simultaneous activation of the secondary nodes with the primary nodes. The enabling PHY layer technology for TC is IC, which is the sole responsibility of the secondary nodes. In this paper, we design a distributed algorithm to achieve TC for multi-hop primary and secondary networks. The main challenge in this algorithm is to ensure that IC is done efficiently (i.e., canceled once by a secondary node) and in a feasible manner (i.e., implementable at the PHY layer). In contrary to a centralized IC algorithm which relies on a global node ordering, we only maintain two local sets for each node to keep track of the node's IC responsibilities. We show how to establish, maintain, and update these two local sets for each node in each iteration of our distributed algorithm. Our distributed algorithm increases the data stream on each active link iteratively based on local computation. Since the nodes in the two local sets of a node directly affect the node's IC responsibility, our algorithm attempts to switch nodes in the two sets if it can improve the IC structure. Although no explicit node ordering is maintained in our distributed algorithm, we prove that our distributed data structure at each node (with the use of two local sets) can be mapped to an explicit global node ordering for IC among all nodes in the network. This guarantees the existences of feasible precoding/decoding vectors at the secondary nodes to achieve our desired IC in the network (i.e., feasibility at the PHY layer). Through simulation study, we show that our distributed algorithm achieves TC between secondary and primary networks and offers competitive throughput performance when compared to a centralized optimization.

ACKNOWLEDGMENTS

This work was supported in part by NSF under Grants 1443889, 1343222, 1102013 and ONR under Grant N00014-15-1-2926. The work of Dr. S. Kompella was supported in part by the ONR. Part of W. Lou's work was completed while she was serving as a Program Director at the NSF. Any opinion, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not reflect the views of the NSF.

REFERENCES

[1] F. Gao, R. Zhang, Y.-C. Liang, and X. Wang, "Design of learning-based MIMO cognitive radio systems," *IEEE Transactions on Vehicular Technology,* vol. 59, no. 4, pp. 1707–1720, May 2010.

[2] S. Geirhofer, L. Tong, and B.M. Sadler, "Dynamic spectrum access in the time domain: Modeling and exploiting white space," *IEEE Communications Magazine,* vol. 45, no. 5, pp. 66–72, May 2007.

[3] A. Goldsmith, S.A. Jafar, I. Maric, and S. Srinivasa, "Breaking spectrum gridlock with cognitive radios: An information theoretic perspective," *Proceedings of the IEEE,* vol. 97, no. 5, pp. 894–914, May 2009.

[4] S. Gollakota and D. Katabi, "Zigzag decoding: Combating hidden terminals in wireless networks," in *Proc. ACM SIGCOMM,* pp. 159–170, Seattle, WA, August 17–22, 2008.

[5] Y.T. Hou, Y. Shi, and H.D. Sherali, "Spectrum sharing for multi-hop networking with cognitive radios," *IEEE Journal on Selected Areas in Commun.,* vol. 26, no. 1, pp. 146–155, Jan. 2008.

[6] Y.T. Hou, Y. Shi, and H.D. Sherali, *Applied Optimization Methods for Wireless Networks,* Cambridge University Press, 2014, ISBN-13: 978-1107018808.

[7] S. Hua, H. Liu, M. Wu, and S.S. Panwar, "Exploiting MIMO antennas in cooperative cognitive radio networks," in *Proc. IEEE INFOCOM,* pp. 2714–2722, Shanghai, China, April. 10-15, 2011.
[8] S.K. Jayaweera, M. Bkassiny, and K.A. Avery, "Asymmetric cooperative communication based spectrum leasing via auctions in cognitive radio networks," *IEEE Trans. on Wireless Commun.,* vol. 10, no. 8, pp. 2716–2724, August 2011.
[9] D. Johnson, Y. Hu, D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," IETF RFC 4728, Feb 2007.
[10] S.-J. Kim and G.B. Giannakis, "Optimal Resource Allocation for MIMO Ad Hoc Cognitive Radio Networks," *IEEE Transactions on Information Theory,* vol. 57, no. 5, pp. 3117–3131, May 2011.
[11] R. Manna, R.H.Y. Louie, Y. Li, and B. Vucetic, "Cooperative spectrum sharing in cognitive radio networks with multiple antennas," *IEEE Trans. on Signal Processing,* vol. 59, no. 11, pp. 5509–5522, Nov. 2011.
[12] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," IETF RFC 3561, July 2003.
[13] H. Rahul, S. Kumar, and D. Katabi, "JMB: scaling wireless capacity with user demands," in *Proc. ACM SIGCOMM,* pp. 235–246, Helsinki, Finland, Aug. 2012.
[14] Y. Shi, J. Liu, C. Jiang, C. Gao, and Y.T. Hou, "A DoF-based link layer model for multi-hop MIMO networks", *IEEE Trans. on Mobile Computing,* vol. 13, no. 7, pp. 1395–1408, July 2014.
[15] O. Simone, I. Stanojev, S. Savazzi, Y. Bar-Ness, U. Spagnolini, and R. Pickholtz, "Spectrum leasing to cooperating secondary ad hoc networks," *IEEE Journal on Selected Areas in Commun.,* vol. 26, no. 1, pp. 203–213, Jan. 2008.
[16] G.S. Smith, "A Direct Derivation of a Single-Antenna Reciprocity Relation for the Time Domain." in *IEEE Trans. on Antennas and Propagation,* vol. 52, no. 6, pp. 1568–1577, June 2004.
[17] A.M. Wyglinski, M. Nekovee, and Y.T. Hou, *Cognitive Radio Communications and Networks: Principles and Practice.* Chapter 12, Academic Press/Elsevier, 2010.
[18] X. Xie, X. Zhang, and K. Sundaresan, "Adaptive feedback compression for MIMO networks," in *Proc. of ACM MobiCom,* pp. 477–488, Miami, FL, Sep. 2013.
[19] X. Yuan, C. Jiang, Y. Shi, Y.T. Hou, W. Lou, S. Kompella, and S.F. Midkiff, "Toward transparent coexistence for multi-hop secondary cognitive radio networks," *IEEE Journal on Selected Areas in Communications,* vol. 33, no. 5, pp. 958–971, May 2015.
[20] X. Yuan, C. Jiang, Y. Shi, Y.T. Hou, W. Lou, and S. Kompella, "Beyond interference avoidance: on transparent coexistence for multi-hop secondary CR networks," in *Proc. IEEE SECON,* pp. 398–405, New Orleans, LA, June 24–27, 2013.
[21] X. Yuan, Y. Shi, Y.T. Hou, W. Lou, S.F. Midkiff, and S. Kompella, "Achieving Transparent Coexistence in a Multi-hop Secondary Network Through Distributed Computation," in *Proc. IEEE IPCCC,* Austin, TX, Dec. 5–7, 2014.
[22] X. Yuan, X. Qin, F. Tian. Y. Shi, Y.T. Hou, W. Lou, S.F. Midkiff, and S. Kompella, "A Distributed Algorithm to Achieve Transparent Coexistence for a Secondary Multi-hop MIMO Network," The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, July 2015. URL: https://www.dropbox.com/s/w65ge63ajgori3j/TR20160117%20 with%20appendix.pdf?dl=0.
[23] X. Yuan, Y. Shi, Y.T. Hou, W. Lou, and S. Kompella, "UPS: A United Cooperative Paradigm for Primary and Secondary Networks," in *Proc. IEEE MASS,* Hangzhou, China, Oct. 14–16, 2013.
[24] R. Zhang and Y.-C. Liang, "Exploiting multi-antennas for opportunistic spectrum sharing in cognitive radio networks," *IEEE Journal of Selected Topics in Signal Processing,* vol. 2, no. 1, pp. 88–102, February 2008.
[25] Y.J. Zhang and A.M.-C. So, "Optimal spectrum sharing in MIMO cognitive radio networks via semidefinite programming," *IEEE Journal on Selected Areas in Communications,* vol. 29, no. 2, pp. 362–373, February 2011.
[26] J. Zhang and Q. Zhang, "Stackelberg game for utility-based cooperative radio network," in *Proc. ACM MobiHoc,* pp. 23–32, New Orleans, LA, USA, May 18–21, 2009.
[27] X. Zhang, K. Sundaresan, M.A. Khojastepour, S. Rangarajan, and K.G. Shin, "NEMOx: scalable network MIMO for wireless networks," in *Proc. ACM MobiCom,* pp. 453-464, Miami, FL, Sep. 2013.
[28] S. Zaks, "Optimal distributed algorithms for sorting and ranking," *IEEE Trans. on Computers,* vol. 5, no. 1, pp. 376–379, April 1985.

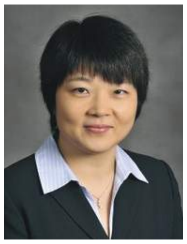
Xu Yuan (S'13–M'16) received his Ph.D. degree in the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA in 2016. His current research interest focuses on algorithm design and optimization for spectrum sharing, coexistence, and cognitive radio networks.

Xiaoqi Qin (S'13) received her B.S. and M.S. degree in Computer Engineering from Virginia Tech in 2011 and 2013, respectively. Since Fall 2013, she has been pursuing her Ph.D. degree in the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA. Her current research interest are algorithm design and cross-layer optimization for wireless networks.

Feng Tian (M'13) received his Ph.D. degree in Signal and Information Processing from Nanjing University of Posts and Telecommunications, Nanjing, China in 2008. He is currently an Associate Professor at the same university. He is a visiting scholar at Virginia Tech, USA from 2013-2015. His research focuses on performance optimization and algorithm design for wireless networks.

Yi Shi (S'02–M'08–SM'13) is a Senior Research Scientist at Intelligent Automation Inc., Rockville, MD, and an Adjunct Assistant Professor at Virginia Tech. His research focuses on optimization and algorithm design for wireless networks and social networks. He has co-organized three IEEE and ACM workshops and has been a TPC member of many major IEEE and ACM conferences. He is an Editor of IEEE Communications Surveys and Tutorials. He authored one book, five book chapters and more than 110 papers on wireless network algorithm design and optimization. He has named an IEEE Communications Surveys and Tutorials Exemplary Editor in 2014. He has a recipient of IEEE INFOCOM 2008 Best Paper Award, IEEE INFOCOM 2011 Best Paper Award Runner-Up, and ACM WUWNet 2014 Best Student Paper Award.

Y. Thomas Hou (F'14) is the Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA. He received his Ph.D. degree from NYU Tandon School of Engineering (formerly Polytechnic Univ.). His current research focuses on developing innovative solutions to complex cross-layer optimization problems in wireless networks. He has published two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). He is the Steering Committee Chair of IEEE INFOCOM conference and a member of the IEEE Communications Society Board of Governors.



Wenjing Lou (F'15) is a professor in the computer science department at Virginia Tech. She received her Ph.D. in Electrical and Computer Engineering from the University of Florida. Her research interests are in the broad area of wireless networks, with special emphases on wireless security and cross-layer network optimization. Since August 2014, she has been serving as a program director at the National Science Foundation. She is the Steering Committee Chair of IEEE Conference on Communications and Network Security (CNS).

Scott F. Midkiff (S'82–M'85–SM'92) is Professor & Vice President for Information Technology and Chief Information Officer at Virginia Tech, Blacksburg, VA. From 2009 to 2012, Prof. Midkiff was the Head of the Bradley Department of Electrical and Computer Engineering at Virginia Tech. From 2006 to 20009, he served as a program director at the National Science Foundation. Prof. Midkiff's research interests include wireless and ad hoc networks, network services for pervasive computing, and cyber-physical systems.

Sastry Kompella (S'04–M'07–SM'12) received his Ph.D. degree in computer engineering from Virginia Tech, Blacksburg, Virginia, in 2006. Currently, he is the Head of Wireless Network Theory section, Information Technology Division at the U.S. Naval Research Laboratory (NRL), Washington, DC. His research focuses on complex problems in cross-layer optimization and scheduling in wireless and cognitive radio networks.