Joint Antenna Allocation and Link Scheduling in FlexRadio Networks

Zhenzhi Qian*, Yang Yang†, Kannan Srinivasan*, Ness B. Shroff*‡
*Department of Computer Science and Engineering, The Ohio State University, Columbus 43210, OH
†Qualcomm Cooperate Research and Development, San Diego 92121, CA
‡Department of Electrical and Computer Engineering, The Ohio State University, Columbus 43210, OH

***Abstract*—FlexRadio, a recent breakthrough in wireless Multi-RF technology, has introduced a new way to unify MIMO and full-duplex into a single framework with a fully flexible design. FlexRadio allows a wireless node to use an arbitrary number of RF chains to support transmission and reception, which makes MIMO and full-duplex subset configurations of FlexRadio. This new architecture has greatly changed the feasibility constraint in wireless networks, which makes the design of high performance MAC layer algorithms even more challenging. First, the RF chain becomes a new resource that needs to be allocated across the network, and the optimal configuration depends not only on the network topology and flow demand, but also on the number of available RF chains at each node. Second, it is not clear how to jointly allocate links and RF chain resources based on the arrival rates and queueing dynamics. In this paper, we introduce a new virtual link model to characterize the feasibility constraint from the perspective of contending RF chain usage. Based on this novel model, a distributed CSMA-like framework is developed to fully leverage the flexibility of RF chain resource allocation.**

I. Introduction

Mobile data traffic is expected to increase by seven times from 2016 to 2021 [1], which poses a data challenge for providing super-high-speed, super-high-capacity wireless communication services. Multi-antenna, a main feature in 5G systems, was first proposed to support Multiple-Input Multiple-Output (MIMO) that could increase the capacity linearly with the number of radio frequency (RF) chains [2]. In MIMO, it is required that all of the RF chains must be used for either transmission or reception with respect to another node. This flow constraint becomes a system bottleneck when multiple users are involved, e.g., in a cellular downlink system. As a result, multi-user (MU) MIMO [3] was developed to support the communication with multiple users simultaneously. However, bi-directional data streams are still not allowed on the same node due to self-interference.

With the help from self-interference cancellation modules, [4] have shown that a wireless node can transmit and receive data streams simultaneously on the same frequency, namely, full-duplex mode is enabled. Based on the physical layer design of full-duplex radios, a node must activate an equal number of RF chains to support simultaneous transmission and reception [5]. In other words, if a node has M RF chains in full-duplex mode, $M/2$ of them are used for transmission while the remaining $M/2$ are designated to be receive RF chains. Compared to the case of MIMO multiplexing, where all M RF chains are used for either transmission or reception, full-duplex does not further increase the capacity region within a pair of nodes or in networks where each node has only 2 RF chains [6]. Even though full-duplex offers the opportunity to enable $M/2$ bi-directional data streams, the same can be achieved by MIMO that supports M unidirectional data streams with time-division for both directions.

To make the design fully flexible, the authors in [5] implemented FlexRadio, a system that enables flexible RF chain resource allocation. FlexRadio allows a single wireless node to use any number of its RF chains for transmission and reception. Specifically, a FlexRadio node can use M_t, M_r and M_n RF chains to transmit, receive or withstand interference as long as $M_t + M_r + M_n \leq M$. The adaptability of FlexRadio enables a wide range of flexible configurations and provides a fundamental improvement in throughput compared to any of the previous Multi-RF technologies as illustrated in Fig. 1. From the network-layer perspective, apart from contending for channel access, data links in FlexRadio networks also need to contend for RF chain usage. This opens a completely new dimension that the scheduler needs to consider, which in turn calls for a redesign of high performance scheduling algorithms for FlexRadio networks, especially when the RF chain resources are limited. ***The key challenge is how to effectively and distributedly choose the optimal configuration which depends on the network topology, flow demands and number of RF chains available at each node?***

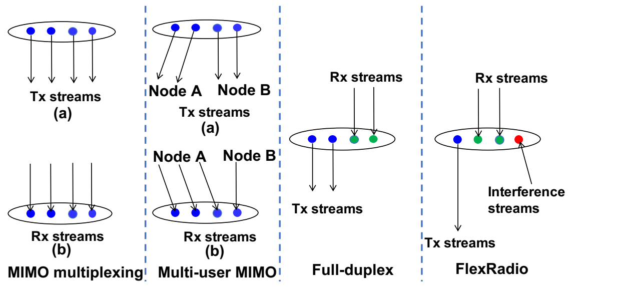

Fig. 1. The development path of Multi-RF technologies

Scheduling algorithms are responsible for dynamically adjusting the transmission configurations, including ON/OFF status of each link, and number of streams on each link in accord with the queueing dynamics of the system, and play a key role in obtaining high throughput and good QoS for different wireless applications. The design of high performance scheduling algorithms is an important problem that has been extensively studied for many years [7]. A famous algorithm called Max-Weight scheduling (MWS) is known to be throughput-optimal, meaning that it can stabilize the network queues under any arrival rates within the capacity region. However, MWS is not practical because of its high computational complexity and centralized design.

A class of Carrier Sensing Multiple Access (CSMA) scheduling algorithms have been investigated in recent works [8, 9] that emulate the functionality of the MWS. CSMA uses a Glauber dynamics-based solution to approximate the optimal solution of the maximum weighted matching problem. CSMA is a simple and distributed scheduling algorithm that achieves throughput-optimality. The authors in [10, 11] provide further enhancements that reduces the delay in CSMA.

Although CSMA is a powerful framework that can solve many problems in wireless scheduling [12, 13], it cannot be directly applied to the scheduling problem in FlexRadio networks. In fact, designing a CSMA-like scheduling algorithm in FlexRadio networks is much more challenging because 1) CSMA relies on a conflict graph to check coexistence criterion, but in FlexRadio networks, concepts like conflict graph do not even exist, because even interfering links may coexist with each other. 2) The feasibility constraint is related to RF chain utilization, which depends on the node's operation mode (is a receiver node or not) and the number of active RF chains at neighboring nodes. Therefore, an important question is: ***Is it possible to design a fully-distributed CSMA-like algorithm that achieves throughput-optimality in FlexRadio networks?***

In this paper, we answer this question in the affirmative by proposing a FlexCSMA scheduling framework. The key contributions of this paper are summarized as follows:

- We introduce a new way to model the RF chain resource allocation problem in FlexRadio networks. To the best of our knowledge, this is the first work to consider the scheduling problem in FlexRadio networks.
- A CSMA-like framework is proposed to dynamically allocate links and RF resources. Throughput-optimality results have been established for the base version (FlexCSMA), distributed version and delayed version (D-FlexCSMA).

The rest of the paper is organized as follows. In Section II, we introduce the network model and feasibility constraint. In Section III, we propose a CSMA-like scheduling algorithm called FlexCSMA that achieves throughput-optimality. In Section IV, we provide a distributed implementation of the FlexCSMA algorithm. In Section V, we design a delayed version of FlexCSMA, which we call D-FlexCSMA to further reduce the delay. We use numerical simulations to validate our theoretical results and compare the performance under different scheduling algorithms in Section VI and make concluding remarks in Section VII.

II. System Model

We consider a wireless network with n FlexRadio nodes where each node $i \in \{1, 2, \cdots, n\}$ has m_i RF chains. Based on the physical layer constraint, each RF chain can be used to either transmit/receive a single intended data stream, or receive and null one interfering stream [6].

A. Network Model

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote the network topology graph, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of data links. There exists a directed data link (i, j) if and only if node j is in node i's transmission range. Let $\mathcal{G}_I = (\mathcal{V}, \mathcal{E}_I)$ denote the interference graph, where $\mathcal{E}_I$ denotes the interference relationship between wireless nodes. For any two wireless nodes i and j, $(i, j) \in \mathcal{E}_I$ if and only if node j is in node i's interference range. For each node $i \in \mathcal{V}$, we define its neighbor set $\mathcal{N}(i) = \{j | (j, i) \in \mathcal{E}_I\}$, note that each node belongs to its own neighbor set, i.e., $i \in \mathcal{N}(i)$. We assume $\mathcal{G}$ and $\mathcal{G}_I$ are both bi-directional and symmetric, and data link set $\mathcal{E}$ is a subset of interference link set $\mathcal{E}_I$.

We consider the high SNR regime, where for each link that utilizes m_t transmit RF chains and m_r receive RF chains, transmission with $\min\{m_t, m_r\}$ *degree-of-freedom* can be achieved [2], and for the rest of this paper we will refer to this as $\min\{m_t, m_r\}$ *multiplexed data streams*.

For any data link $l = (i, j) \in \mathcal{E}$, there could be up to $C_l = \min\{m_i, m_j\}$ *multiplexed data streams* on this data link. We use a virtual link $l_v = (i, j, v)$, $1 \leq v \leq C_l$ to denote the v^{th} data stream. Throughout this paper, virtual links $\{l_v\}_{v=1}^{C_l}$ defined on the same data link l are treated equally, and v is used only for indexing purpose. If link l is using the configuration that supports r *multiplexed data streams*, then we should have r active virtual links (i.e., any r out of C_l virtual links) on link l. In other words, let $\mathcal{S}_l$ denote the set of all active virtual links on link l, then the link rate r_l is equal to $\sum_{v=1}^{C_l} \mathbf{1}_{\{l_v \in \mathcal{S}_l\}}$. For each virtual link l, we use $s(l_v)$, $d(l_v)$ to denote the transmitter node, receiver node of the virtual link l_v (in this case, $s(l_v) = i$, $d(l_v) = j$). We further denote the set of all possible virtual links as

$$\mathcal{E}_v = \{l_v = (i, j, v) | l = (i, j) \in \mathcal{E}, 1 \leq v \leq C_l\}.$$

Remark 1: To activate each virtual link, we need to activate one transmit RF chain on the transmitter node and one receive RF chain on the receiver node. Virtual links are not tied with any specific RF chains – whenever a virtual link were to be activated, a random available pair of transmit and receive RF chains is picked.

For the wireless networks with FlexRadio nodes, RF chains distributed at each node become a new type of resource that needs to be scheduled and coordinated. Data link l could activate more virtual links to reach a higher link rate at the cost of more RF chain resource utilization, which leaves less room for the activation of its neighboring links. In the next section, we will provide a precise characterization of the feasibility constraint, capturing such conflicts between data links.

B. Feasibility Constraint

A schedule $\mathcal{S}$ is a subset of virtual link set $\mathcal{E}_v$. For ease of presentation, we use x_{l_v} to denote the virtual link activation status with $x_{l_v} = 1$ if $l_v \in \mathcal{S}$ and $x_{l_v} = 0$ otherwise. For a given schedule, we first define a receiver indicator vector $\mathbf{e}$, where its i^{th} element e_i denotes the number of virtual links that use node i as the receiver. Formally, $e_i = \sum_{d(l_v)=i} x_{l_v}$ for $1 \leq i \leq n$.

Definition 1: A node i is in receiving (Rx) mode if $e_i > 0$. If $e_i = 0$, then node i is not a receiver for any active virtual link, node i is either inactive or only in transmitting (Tx) mode. With the support of full-duplex, it is possible for node i to be in both Tx and Rx mode simultaneously.

Next, we use a vector $\mathbf{a}$ to indicate each node's RF chain utilization state. The i^{th} element a_i denotes the number of occupied RF chains on node i. Given vectors $\mathbf{x}$ and $\mathbf{e}$, a_i is given by:

$$a_i = \begin{cases} \sum\limits_{i \in \{s(l_v), d(l_v)\}} x_{l_v} + \sum\limits_{\substack{d(l_v) \neq i, \\ s(l_v) \in \mathcal{N}(i) \setminus \{i\}}} x_{l_v}, \text{ if } e_i > 0, \\ \sum\limits_{s(l_v)=i} x_{l_v}, \text{ otherwise.} \end{cases} \tag{1}$$

For each node that is in Rx mode, one RF chain is required to support each active transmission or reception. In addition, in order to decode the intended data streams successfully, node i needs to use one additional RF chain to withstand each interference stream from its neighboring nodes. For all other nodes that are not in Rx mode, we do not need to do interference management, in which case, a_i simply counts the number of data streams that node i should transmit (if there is any). Note that if node i is in idle state (not in Tx mode or Rx mode), no RF chain is utilized and $a_i = 0$.

Depending on the node state (Rx mode or non-Rx mode), the computation of $\mathbf{a}$ is completely different as evident from (1). Whenever a virtual link l_v is added to a schedule, it may change the mode of the receiver node $d(l_v)$. Prior to the addition of the new virtual link, if node $d(l_v)$ is not in Rx mode, then $a_{d(l_v)}$ only needs to account for all transmitting data streams originated from node $d(l_v)$. Now, after we add link l_v to the existing schedule, node $d(l_v)$ is in Rx mode. In order to cancel interference streams from neighboring links, we have to allocate one RF chain for each interfering data stream. Similarly, when we delete a link l_v from the existing schedule, we may also change the state of node $d(l_v)$ (from Rx mode to non-Rx mode). In the non-Rx mode, it is no longer necessary to allocate additional dimensions of RF resources for interference nulling, but we still need to keep track of the number of "overhearing" interference streams to prepare for any potential state change and the update of RF chain utilization vector $\mathbf{a}$.

We use a vector $\mathbf{h}$ to denote the number of data streams that node i hears (including both its intended data streams and the interfering streams):

$$h_i = \sum_{l_v} x_{l_v} \mathbf{1}_{\{s(l_v) \in \mathcal{N}(i) \setminus \{i\}\}}, \forall i \in \mathcal{V}, \tag{2}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

A feasible schedule $\mathcal{S}$ is a schedule such that each data link l can successfully receive $\sum_{v=1}^{C_l} x_{l_v}$ data streams. A feasible schedule $\mathcal{S}$ must satisfy that each wireless node has enough RF chain resources to handle transmitting, receiving data streams and nulling interference. In particular, a schedule $\mathcal{S}$ is feasible if the following constraint is satisfied:

$$a_i^{\mathcal{S}} \leq m_i, \forall i \in \mathcal{V}. \tag{3}$$

Constraint (3) says that the number of RF chains needed to support $\mathcal{S}$ should not exceed the number of RF chains on each node. This represents the physical layer limitation that one RF chain can only be used to transmit one data stream or receive one data stream.

For each data link $l = (i, j)$, there is a queue storing all the data packets to be transmitted from node i to node j. We assume the single-hop traffic model, where data packets leave the network immediately after they are received by the receiver nodes. Consider a slotted system, and denote $Q_l(t)$ as the queue length of data link l in time-slot t. Based on the model, we have the following queue-length evolution equation:

$$Q_l(t) = \left[Q_l(t-1) + A_l(t) - \sum_{v=1}^{C_l} x_{l_v}(t)\right]^+, \tag{4}$$

where $(x)^+ = \max\{x, 0\}$, and $A_l(t)$ is the number of new data requests from node i to node j in time-slot t.

Given a feasible schedule $\mathcal{S}$, the data rate (number of *multiplexed data streams*) $r_l^{\mathcal{S}}$ on data link l is given by $r_l^{\mathcal{S}} = \sum_{v=1}^{C_l} x_{l_v}$. Let $\mathbb{M}$ denote the set of all possible feasible schedules and $\mathbb{D}$ denote the set of feasible rate vectors of all schedules in $\mathbb{M}$, i.e., $\mathbb{D} = \{\mathbf{r}^{\mathcal{S}} | \mathcal{S} \in \mathbb{M}\}$.

C. Objective

In this paper, we mainly focus on the single-hop traffic and our goal is to achieve the capacity region of the network. A scheduling algorithm governs which antenna configurations to be used and which data links to be activated in each time-slot t. In this problem, the scheduling problem boils down to select a feasible (virtual) link schedule in each time-slot. The capacity region is defined as the set of all arrival rate vectors $\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_{|\mathcal{E}|}\}$ such that there exists a scheduling algorithm that can stabilize the queueing network. In other words, given the stochastic arrival process, the queueing network behaves as a discrete-time Markov chain (DTMC), and stability refers to the positive recurrent property of this Markov chain. It is well-known in [14] that the capacity region is given by:

$$\Lambda = \{\lambda | \lambda \preceq \theta \text{ for some } \theta \in Co(\mathbb{D})\}. \tag{5}$$

where $Co(\mathbb{D})$ is the convex hull of $\mathbb{D}$.

Definition 2: A scheduling algorithm is throughput-optimal if it can stabilize all arrival rates inside Λ.

Our objective is to find a low-complexity joint antenna allocation and link scheduling algorithm that achieves throughput-optimal performance.

III. FlexCSMA: A CSMA Algorithm for Scheduling FlexRadio Networks

In this section, we aim to develop a CSMA-like algorithm that provides provably throughput-optimal performance. Q-CSMA [9] algorithms are promising because they incur low-complexity and can be shown to maximize throughput under certain assumptions. However, the traditional Q-CSMA algorithms cannot be directly applied to this problem for the following reasons:

- The traditional Q-CSMA relies on the construction of a conflict graph which characterizes the interference relationship between different links. However, the constraint captured in (3), is node-oriented, and thus there is no clear conflict graph between virtual links in FlexRadio networks. Whether two virtual links can be scheduled simultaneously or not depends not only on the interference relationship, but also on the number of available RF chains and node state (Rx or non-Rx mode).
- The feasibility constraint is no longer binary, the coexistence criterion may depend on the link activation decisions of other virtual links. A new design of the decision schedule is needed to satisfy Markov property.

Fortunately, with the introduction of a novel two-stage procedure for each iteration, we are still able to develop a CSMA-like joint antenna allocation and link scheduling algorithm and restore the throughput-optimality. We divide each time slot t into two phases, the first phase is called *control slot*, which is used to generate a feasible transmission schedule $\mathcal{S}(t) \in \mathbb{M}$ and its corresponding link state vector $\mathbf{x}(t)$. Then, in the second phase, namely *data slot*, each node will use the configuration specified by $\mathbf{x}(t)$ to randomly assign RF chains for the transmission.

A. Decision Schedule

The algorithm first selects a set of virtual links $\mathcal{M}(t)$ in each time-slot t, such that for any two virtual links $l_v, l'_u \in \mathcal{M}(t)$, we have:

$$\mathcal{N}(s(l_v)) \cap \mathcal{N}(s(l'_u)) = \emptyset, \tag{6}$$

indicating that the transmitters in $\mathcal{M}(t)$ do not have a common neighbor node. $\mathcal{M}(t)$ is also called the *decision schedule* in time-slot t. The intuition behind the decision schedule requirement (6) is to make sure the CSMA-like algorithm satisfies Markov property.

Recall that in Q-CSMA [9], after $k-1$ links have made their activation decisions, the coexistence criterion such that activating the k^{th} link from $\mathcal{M}(t)$ does not incur any interference remains unchanged regardless of the activation decisions of the previous $k-1$ links. In other words, each link in $\mathcal{M}(t)$ is able to check its coexistence criterion based on the previous schedule $\mathcal{S}(t-1)$.

In this problem, we aim to emulate the same role of the decision schedule in Q-CSMA. Consider any decision schedule $\mathcal{M}(t)$ that satisfies (6), the coexistence criterion of each virtual link $l_v \in \mathcal{M}(t)$ is related to the RF chain utilization states of all nodes in its neighborhood $\mathcal{N}(s(l_v))$ to make sure the reception is successful. On the other hand, the link activation decision of l_v may also change the RF chain utilization states of nodes from the same set $\mathcal{N}(s(l_v))$. Based on (6), it is clear that the link activation decision of any other virtual link $l'_u \in \mathcal{M}(t)$ will not change the RF chain utilization states of nodes in $\mathcal{N}(s(l_v))$. Hence, the coexistence criterion of l_v remains unchanged regardless of the other virtual links' activation decisions, and the following lemma holds.

Lemma 1: $\mathcal{M}(t)$ is also a feasible schedule.

Remark 2: An alternative decision schedule is to use a set of virtual links that can coexist with each other. However, it does not work because it violates the Markov property.

A simple example is shown in Fig. 2, we consider a network with 4 nodes $\{A, B, C, D\}$ and number of RF chains $\{4, 3, 3, 5\}$, respectively. Now, given the existing schedule $\mathcal{S}(t-1) = \{(A, C, 1), (B, A, 1), (B, D, 1)\}$, only one virtual link from the decision schedule $\mathcal{M}(t) = \{(A, C, 2), (B, A, 2)\}$ is allowed to be included in $\mathcal{S}(t)$. Adding both of them will cause collision (lack of RF chains) at node A. Hence, there is a dependency between the activation decisions of virtual links in $\mathcal{M}(t)$, which violates the Markov property.

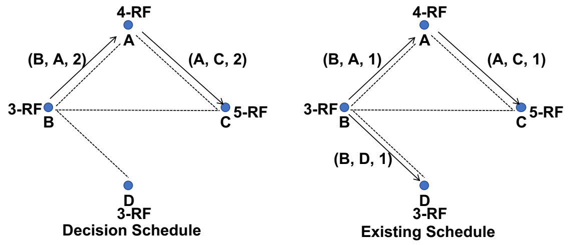

Fig. 2. A counter example.

B. Overview of the FlexCSMA Algorithm

Let $\mathbb{M}_0 \subseteq \mathbb{M}$ denote the set of all decision schedules. In the control slot, the network randomly selects a decision schedule $\mathcal{M}(t)$ with non-negative probability $\alpha(\mathcal{M}(t))$. $\alpha : \mathbb{M}_0 \to [0, 1]$ is a probability measure with $\sum_{\mathcal{M}(t)\in\mathbb{M}_0} \alpha(\mathcal{M}(t)) = 1$. The transmission schedule $\mathcal{S}(t)$ and link state vector $\mathbf{x}(t)$ are determined as follows. For each virtual link $l_v \in \mathcal{M}(t)$, we update vectors $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$ as if we already include virtual link l_v in the current schedule $\mathcal{S}(t)$ (omit this step if virtual link $l_v \in \mathcal{S}(t-1)$). If vector $\mathbf{a}$ satisfies the feasibility constraint (3), we include l_v in the current schedule with probability p_{l_v} and deactivate l_v with probability $\overline{p}_{l_v} = 1 - p_{l_v}$. If constraint (3) does not hold for vector $\mathbf{a}$, virtual link l_v will not be included in the current schedule $\mathcal{S}(t)$, and all vectors $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$ will be recovered to the original value. For all other virtual links $l_v \notin \mathcal{M}(t)$, the link state remains the same as in the previous time-slot, i.e., $x_{l_v}(t) = x_{l_v}(t-1)$. In the subsequent data slot, we activate virtual link l_v if $l_v \in \mathcal{S}(t)$, i.e., $x_{l_v}(t) = 1$.

Algorithm 1 FlexCSMA Scheduling Algorithm

```
1:  1. In the control slot, randomly select a decision schedule
       $\mathcal{M}(t)$ from $\mathbb{M}_0$ with probability $\alpha(\mathcal{M}(t))$.
2:  for each virtual link $l_v \in \mathcal{M}(t)$ do
3:      if AddCheck($l_v$)=True then
4:          a. $x_{l_v}(t) = 1$ with probability $p_{l_v}$;
5:          b. $x_{l_v}(t) = 0$, Del($l_v$) with probability $1 - p_{l_v}$;
6:      else
7:          $x_{l_v}(t) = 0$;
8:          Del($l_v$);
9:  for each link $l_v \notin \mathcal{M}(t)$ do
10:     $x_{l_v}(t) = x_{l_v}(t-1)$;
11: 2. In the data slot, if $x_{l_v}(t) = 1$, include $l_v$ in $\mathcal{S}(t)$.
```

Unlike the traditional Q-CSMA framework, checking link coexistence is no longer easy and intuitive. There is no counterpart of conflict graph, instead, we use two important procedures, namely *AddCheck(l_v)* and *Del(l_v)* to check coexistence and update vectors $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$.

C. Add Link and Check Coexistence

The purpose of the procedure *AddCheck(l_v)* is to 1) Update vectors $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$ as if l_v has already been included in the current schedule $\mathcal{S}(t)$. 2) Use feasibility constraint (3) to check if the combining schedule $\mathcal{S}(t-1) \cup \{l_v\}$ is feasible.

If virtual link l_v appears in the previous schedule, i.e., $x_{l_v}(t-1) = 1$, we have $\mathcal{S}(t-1) \cup \{l_v\} = \mathcal{S}(t-1)$. The combining schedule is always feasible and there is no need to update vectors $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$.

For each virtual link l_v such that $l_v \notin \mathcal{S}(t-1)$, adding l_v only affects nodes that are within the interference range of the source node $s(l_v)$, i.e., nodes belong to $\mathcal{N}(s(l_v))$. In particular, the receiver node $d(l_v)$ is the only one that changes receiver indicator value e_i, which could result in a state change from non-Rx mode to Rx mode. If the state change does happen, the RF chain utilization value $a_{d(l_v)}$ is added by the previous link hearing value $h_{d(l_v)}$ and data rate 1 of virtual link l_v. If node $d(l_v)$ is already in Rx mode in the previous time-slot, then the hearing information has already been considered in the previous $a_{d(l_v)}$, we just simply add $a_{d(l_v)}$ by link rate 1. We can apply the same argument to any other neighboring nodes $i \in \mathcal{N}(s(l_v))$ (node i is either the source node of l_v or $e_i > 0$) to update the value of a_i. Each node i that belongs to $\mathcal{N}(s(l_v))$ except for $s(l_v)$ itself hears one more data stream and h_i is updated accordingly. Finally, given the updated vector $\mathbf{a}$, we check the feasibility constraint (3) and return the coexistence result of virtual link l_v against the previous schedule $\mathcal{S}(t-1)$.

Algorithm 2 Add Link and Check Coexistence

```
1: procedure AddCheck($l_v$)
2:     if $x_{l_v}(t-1) \neq 1$ then
3:         for each node $i \in \mathcal{N}(s(l_v))$ do
4:             if $i = d(l_v)$ then
5:                 if $e_i = 0$ then
6:                     $a_i = a_i + h_i + 1$;
7:                 else
8:                     $a_i = a_i + 1$;
9:                 $e_i = e_i + 1$;
10:            else
11:                if $i = s(l_v)$ or $e_i > 0$ then
12:                    $a_i = a_i + 1$;
13:            if $i \neq s(l_v)$ then
14:                $h_i = h_i + 1$;
15:        for each node $i \in \mathcal{N}(s(l_v))$ do
16:            if $a_i > m_i$ then
17:                return False;
18:    return True;
```

D. Delete Link and Update States

After FlexCSMA algorithm invokes *AddCheck(l_v)*, we already update vectors $\mathbf{a}, \mathbf{h}$ and $\mathbf{e}$ as if l_v is included in the current schedule. However, if we decide to delete l_v from the current schedule, we must recover the original vectors $\mathbf{a}, \mathbf{h}$ and $\mathbf{e}$. The purpose of the procedure *Del(l_v)* is to deactivate l_v from the current schedule and recover vectors $\mathbf{a}, \mathbf{h}$ and $\mathbf{e}$.

Algorithm 3 Delete Link and Recover States

```
1: procedure Del($l_v$)
2:     for each node $i \in \mathcal{N}(s(l_v))$ do
3:         if $i = d(l_v)$ then
4:             $e_i = e_i - 1$;
5:             if $e_i = 0$ then
6:                 $a_i = a_i - h_i$;
7:             else
8:                 $a_i = a_i - 1$;
9:         else
10:            if $i = s(l_v)$ or $e_i > 0$ then
11:                $a_i = a_i - 1$;
12:        if $i \neq s(l_v)$ then
13:            $h_i = h_i - 1$;
```

Remark 3: Let $\mathbf{a}(t), \mathbf{h}(t)$ and $\mathbf{e}(t)$ take the value of the corresponding vector $\mathbf{a}, \mathbf{h}$ and $\mathbf{e}$ at the end of each time-slot t. The value of vectors $\mathbf{a}(t), \mathbf{h}(t)$ and $\mathbf{e}(t)$ are actually determined by $\mathbf{x}(t)$. Instead of recomputing these vectors in each time-slot, we use $\mathbf{a}$, $\mathbf{h}$ and $\mathbf{e}$ to keep track of these vectors from time to time.

E. Throughput Analysis

In this section, we analyze the throughput of Algorithm 1.

Lemma 2: If $\mathcal{S}(t-1) \in \mathbb{M}$ and $\mathcal{M}(t) \in \mathbb{M}_0$, then $\mathcal{S}(t) \in \mathbb{M}$.

Proof: Note that $\mathcal{S}(t) \in \mathbb{M}$ if constraint (3) is satisfied. Consider any node $i \in \mathcal{V}$, and the set of virtual links that could affect node i's RF chain utilization state $\mathcal{L}(i) = \{l_v | i \in \mathcal{N}(s(l_v))\}$. Then $\mathcal{S}(t) \cap \mathcal{L}(i)$ is the set of active virtual links that occupy node i's RF chains under schedule $\mathcal{S}(t)$. Among all virtual links in this set, there is at most one virtual link l_m that belongs to $\mathcal{M}(t)$. The uniqueness is guaranteed by the property of the decision schedule requirement (6).

If such virtual link l_m does not exist, then all virtual links in $\mathcal{S}(t) \cap \mathcal{L}(i)$ are not selected in $\mathcal{M}(t)$. From Algorithm 1, a virtual link l_m could change its link state from $x_{l_m}(t-1)$ to $x_{l_m}(t)$ if and only if virtual link $l_m \in \mathcal{M}(t)$. Therefore, all virtual links in $\mathcal{S}(t) \cap \mathcal{L}(i)$ have already appeared in the previous schedule $\mathcal{S}(t-1)$. Since the schedule $\mathcal{S}(t-1)$ is feasible, we have $a_i^{\mathcal{S}(t)} \leq a_i^{\mathcal{S}(t-1)} \leq m_i$.

On the other hand, if there exists a unique virtual link l_m that belongs to $\mathcal{S}(t) \cap \mathcal{L}(i) \cap \mathcal{M}(t)$. According to Algorithm 1, virtual link l_m could be included in $\mathcal{S}(t)$ only if $\mathbf{a}^{\mathcal{S}(t-1) \cup \{l_m\}}$ satisfies constraint (3). Thus, we have $a_i^{\mathcal{S}(t-1) \cup \{l_m\}} \leq m_i$. We only need to show that the sets of active virtual links that occupy node i's antennas under schedules $\mathcal{S}(t-1) \cup \{l_m\}$ and $\mathcal{S}(t)$ are the same, i.e., $\mathcal{S}(t) \cap \mathcal{L}(i) = (\mathcal{S}(t-1) \cup \{l_m\}) \cap \mathcal{L}(i)$.

Applying the distributive law with $\mathcal{L}(i) \cap \{l_m\} = \{l_m\}$, it suffices to show $\mathcal{S}(t) \cap \mathcal{L}(i) = (\mathcal{S}(t-1) \cap \mathcal{L}(i)) \cup \{l_m\}$. First, note that any virtual link in $(\mathcal{S}(t) \cap \mathcal{L}(i)) \backslash \{l_m\}$ does not belong to $\mathcal{M}(t)$, so all these virtual links have already appeared in $\mathcal{S}(t-1) \cap \mathcal{L}(i)$. We have $\mathcal{S}(t) \cap \mathcal{L}(i) \subseteq (\mathcal{S}(t-1) \cap \mathcal{L}(i)) \cup \{l_m\}$ by taking the union of $\{l_m\}$ on both sides. Next, let us assume there exists a virtual link $l'_u \in (\mathcal{S}(t-1) \cap \mathcal{L}(i)) \cup \{l_m\}$ such that $l'_u \notin \mathcal{S}(t) \cap \mathcal{L}(i)$. Obviously, since $l_m \in \mathcal{S}(t) \cap \mathcal{L}(i)$, we have $l'_u \neq l_m$ and $l'_u \in \mathcal{S}(t-1) \cap \mathcal{L}(i)$. In addition, we know that l'_u is not included in $\mathcal{S}(t)$, otherwise $l'_u \in \mathcal{S}(t) \cap \mathcal{L}(i)$. Note that $l'_u \in \mathcal{S}(t-1)$ but $l'_u \notin \mathcal{S}(t)$, virtual link l'_u has changed its link state from 1 (*active*) to 0 (*inactive*), so virtual link l'_u must belong to $\mathcal{M}(t)$. Therefore, we have already found two different links l_m and l'_u in $\mathcal{M}(t)$, such that $l_m, l'_u \in \mathcal{L}(i)$ and $i \in \mathcal{N}(s(l_m)) \cap \mathcal{N}(s(l'_u))$. This contradicts with the property of the decision schedule (6), there is no such link l'_u and we have derived the desired result. ■

It is clear that $\mathbf{a}(t-1), \mathbf{h}(t-1)$ and $\mathbf{e}(t-1)$ are all determined by $\mathbf{x}(t-1)$. Algorithm 1 uses $\mathbf{x}(t-1), \mathbf{a}(t-1), \mathbf{h}(t-1), \mathbf{e}(t-1)$ and a randomly selected set $\mathcal{M}(t)$ to generate $\mathbf{x}(t)$. Thus, $\mathcal{S}(t)$ only depends on $\mathcal{S}(t-1)$ and $\mathcal{M}(t)$. $\mathcal{S}(t)$ satisfies Markov property and it evolves as a discrete-time Markov chain (DTMC).

Lemma 3: A feasible schedule $\mathcal{S} \in \mathbb{M}$ can make a feasible transition to another state $\mathcal{S}' \in \mathbb{M}$ if and only if $\mathcal{S} \cup \mathcal{S}' \in \mathbb{M}$ and there exists a decision schedule $\mathcal{M} \in \mathbb{M}_0$ such that

$$\mathcal{S} \triangle \mathcal{S}' = (\mathcal{S} \backslash \mathcal{S}') \cup (\mathcal{S}' \backslash \mathcal{S}) \subseteq \mathcal{M}, \tag{7}$$

and the transition probability from $\mathcal{S}$ to $\mathcal{S}'$ is:

$$P(\mathcal{S},\mathcal{S}') = \sum_{\mathcal{S} \triangle \mathcal{S}' \subseteq \mathcal{M}} \alpha(\mathcal{M}) \left(\prod_{l_{v_1} \in \mathcal{S} \backslash \mathcal{S}'} \overline{p}_{l_{v_1}} \right) \left(\prod_{i_{v_2} \in \mathcal{S}' \backslash \mathcal{S}} p_{i_{v_2}} \right) \left(\prod_{j_{v_3} \in \mathcal{M} \cap (\mathcal{S} \cap \mathcal{S}')} p_{j_{v_3}} \right) \left(\prod_{q_{v_5} \in \mathcal{M} \backslash (\mathcal{S} \cup \mathcal{S}') \backslash \mathcal{I}((\mathcal{S} \cup \mathcal{S}'))} \overline{p}_{q_{v_5}} \right). \tag{8}$$

where $\mathcal{I}(S)$ denote the set of virtual links that do not belongs to $\mathcal{S}$ and cannot coexist with $\mathcal{S}$.

The proof is omitted due to space limitation, we only present the following lemma, which is essential to make sure the Markov Chain with the transition probability (8) is reversible.

Lemma 4: $\mathcal{M} \cap (\mathcal{I}(\mathcal{S} \cup \mathcal{S}') \backslash \mathcal{I}(\mathcal{S})) = \emptyset$.

Proof: Assume there exists a virtual link $l_v \in \mathcal{M}$, such that $l_v \in \mathcal{I}(\mathcal{S} \cup \mathcal{S}')$ but $l_v \notin \mathcal{I}(\mathcal{S})$. According to the definition of set $\mathcal{I}$, we know $l_v \notin \mathcal{S}$, $l_v \notin \mathcal{S}'$, l_v could coexist with $\mathcal{S}$ but cannot coexist with $\mathcal{S} \cup \mathcal{S}'$. Since l_v is selected by the decision schedule $\mathcal{M}$, it is the only virtual link that could affect the RF chain utilization state for nodes in $\mathcal{N}(s(l_v))$. As a result, in schedule $\mathcal{S} \cup \mathcal{S}'$, nodes in $\mathcal{N}(s(l_v))$ do not change its vector $\mathbf{a}$ from $\mathcal{S}$. Therefore, l_v should still coexist with $\mathcal{S} \cup \mathcal{S}'$, which is a contradiction. ■

Proposition 1: For any virtual link $l_v \in \mathcal{E}_v$, let the activation probability $p_{l_v} = \frac{\exp(w_{l_v}(t))}{\exp(w_{l_v}(t))+1}$, where $w_{l_v}(t)$s are appropriate functions of queue-length $Q_l(t)$ on link l. Then Algorithm 1 is throughput-optimal.

We omit the proof since it is standard under the time-scale separation assumption. In fact, if we define $w_{l_v}(t) = w_l(t)$ for all v, then the steady-state probability of selecting a schedule $\mathcal{S}$ is proportional to the sum weight of virtual links $\sum_{l_v \in \mathcal{S}} w_{l_v}(t) = \sum_{l \in \mathcal{E}} w_l(t) r_l^{\mathcal{S}}$, which approximates the solution for the Max-Weight problem under multi-rate scenarios. Some appropriate functions for $w_{l_v}(t)$ are: $w_{l_v}(t) = \log(\gamma Q_l(t))$, $w_{l_v}(t) = \log\log(Q_l(t) + e)$ and $w_{l_v}(t) = \log(1 + Q_l(t))^{1-\gamma}$ for a small $\gamma > 0$.

F. An Alternative Definition of the Virtual Link

A natural way to define the virtual link is based on the link configuration as in [13]. Specifically, given a data link l, virtual link l_v represents the link configuration on data link l with rate v, for $v = 1, 2, \cdots, C_l$. In this virtual link model, we can show that by slightly changing our existing FlexCSMA framework, the resulting scheduling algorithm is still throughput-optimal under the time-scale separation assumption.

Now, the key question is: *How long does it take to converge to the steady-state?* In FlexCSMA, each virtual link only represents one data stream. The scheduler can activate or deactivate at most one virtual link l_v at a time to change the link rate of data link l. As time goes on, the link rate will gradually accumulate or depreciate based on the queueing dynamics. Whereas in this approach, link configurations are changed as we change the schedule, hence, the link rate transition is much more dramatic. One may expect it to converge much faster due to its fast response and high efficiency. Surprisingly, it is the other way around.

Here is a simple example. Suppose there is only one link l, the transmitter and receiver both have 10 RF chains so that $C_l = 10$. Consider a feasible arrival rate of 9.9, FlexCSMA will activate more and more virtual links with high probability. In the end, link schedule will have all 10 virtual links activated for most of time. On the other hand, in this approach, let us assume the first virtual link activated by this approach is l_8, which provides link rate of 8. Now if the decision schedule happens to select l_8, then with high probability, l_8 will remain active. Hence, the schedule will stuck in the state $\{l_8\}$ for a long time before it becomes empty again, and the decision schedule may not necessarily select l_{10} even after the scheduler decides to deactivate l_8.

In general, if the queue-length of Q_l becomes very large and the scheduler has the incentive to increase the link rate from v_1 to v_2. In this approach, there is no direct transition from l_{v_1} to l_{v_2} on link l. Virtual link l_{v_1} must be deactivated first to make room for the rate increase, which does not make sense from the perspective of virtual link l_{v_1}. As a result, the schedule will keep l_{v_1} for a long time and take a much longer time to converge to the steady-state.

IV. Distributed Implementation of the FlexCSMA Algorithm

In this section, we propose a distributed implementation of the FlexCSMA scheduling algorithm. We divide the control slot into control mini-slots. At the beginning of each time-slot t, each data link l randomly selects an index v to generate a candidate virtual link l_v. Virtual link l_v then selects a random backoff time T_l uniformly from the interval $[0, W-1]$ and wait for T_l control mini-slots. A node is **MARKED** if it has transmitted an **INTENT** message without collision or it has received a **MARK** message so far. If any node in $\mathcal{N}(i)$ is **MARKED** before $(T_l+1)^{th}$ control mini-slot, then adding virtual link l_v to the decision schedule violates the requirement (6) and virtual link l_v will not be included in the decision schedule $\mathcal{M}(t)$. Otherwise, if all nodes in $\mathcal{N}(i)$ are not **MARKED** before $(T_l+1)^{th}$ control mini-slot, virtual link l_v will transmit **INTENT** message and if there is no collision at any node in $\mathcal{N}(i)$, a **MARK** message will be broadcasted, and we will apply the same procedure in Algorithm 1 to determine the link state of virtual link l_v in time-slot t.

Algorithm 4 Distributed FlexCSMA Algorithm (At data link l in time-slot t)

```
1: Data link $l = (i, j)$ randomly selects an index $v$ uniformly in $[1, C_l]$ to generate a virtual link $l_v = (i, j, v)$ and set $x_{l_k}(t) = x_{l_k}(t-1)$, $\forall 1 \le k \le C_l, k \ne v$.
2: Data link $l$ selects a random backoff time $T_l$ uniformly in $[0, W-1]$ and waits for $T_l$ control mini-slots. If the receiver node $j$ hears an **MARK** message before the $(T_l+1)^{th}$ control mini-slot, node $j$ is **MARKED**.
3: IF any node in the set $\mathcal{N}(i)$ is **MARKED** before the $(T_l+1)^{th}$ control mini-slot, virtual link $l_v$ will not be included in the decision schedule $\mathcal{M}(t)$ and the data link $l$ will no longer transmit an **INTENT** message any more. Set $x_{l_v}(t) = x_{l_v}(t-1)$.
4: IF all nodes in the set $\mathcal{N}(i)$ are not **MARKED** before the $(T_l+1)^{th}$ control time-slot, data link $l$ will broadcast an **INTENT** message at the beginning of the $(T_l+1)^{th}$ control mini-slot.
     • If there is a collision or multiple **INTENT** messages are heard by a certain node in $\mathcal{N}(i)$ in $(T_l+1)^{th}$ control mini-slot, Virtual link $l_v$ is not included in the decision schedule $\mathcal{M}(t)$, and set $x_{l_v}(t) = x_{l_v}(t-1)$.
     • Otherwise, set the sender node $i$ as **MARKED** and broadcast a **MARK** message to $\mathcal{N}(i)$, virtual link $l_v$ will be included in $\mathcal{M}(t)$ and determine its state as follows:
       **If** AddCheck($l_v$)=True
          a. $x_{l_v}(t) = 1$ with probability $p_{l_v}$;
          b. $x_{l_v}(t) = 0$, Del($l_v$) with probability $1 - p_{l_v}$;
       **Else**
          $x_{l_v}(t) = 0$, Del($l_v$);
5: IF $x_{l_v}(t) = 1$, sender node $i$ and receiver node $j$ will randomly allocate one available RF chain to transmit one data stream on link $l$.
```

Proposition 2: Algorithm 4 is throughput-optimal if $W \ge 2$ and the activation probability takes the form of $p_{l_v} = \frac{\exp(w_{l_v}(t))}{\exp(w_{l_v}(t))+1}$ for some appropriate $w_{l_v}(t)$s.

We can extend the proof of Proposition 1 here.

V. Delay Reduction of the FlexCSMA Algorithm

Despite the merit that CSMA is a simple and distributed algorithm that can achieve throughput-optimality, empirical evidence [10] has found that CSMA often results in large queue lengths. Hence, a key problem in CSMA scheduling is how to reduce the delay. Queueing delay depends not only on arrival/service rate but also on the correlation of service over time. In Q-CSMA, if two links are contending for channel access, one link has to wait for the other link (if it is active) to release the channel first before it can be active in the next time-slot. Since there is no direct transition from one active link to the other, the scheduler spends a non-negligible time on the switch between schedules which wastes quite a few transmission opportunities during the transient schedules. In FlexCSMA, the delay problem becomes more severe as one link may need to wait for the other links to release RF chain resources to start ramping up its own link rates. Also, the release and ramping up process may take at most $\min\{m_{s(l)}, m_{d(l)}\}$ time-slots to complete on link l, where $m_{s(l)}$ and $m_{d(l)}$ is the number of RF chains on the transmitter and receiver node, respectively. In this paper, we borrow the idea from [11] to develop a delayed version of FlexCSMA, we call D-FlexCSMA, which takes T-step-back state of vectors $\mathbf{a}, \mathbf{e}, \mathbf{h}$ to determine the next schedule $\mathcal{S}(t)$. In this way, the system behaves as if T independent chains take turn to decide the next schedule. The formal description of D-FlexCSMA is shown in Algorithm 5.

Algorithm 5 D-FlexCSMA Scheduling Algorithm

```
1: 1. In the control slot, randomly select a decision schedule $\mathcal{M}(t)$ from $\mathbb{M}_0$ with probability $\alpha(\mathcal{M}(t))$.
2: **for** each virtual link $l_v \in \mathcal{M}(t)$ **do**
3:     **if** AddCheck($l_v$, $t \mod T$)=True **then**
4:         a. $x_{l_v}(t) = 1$ with probability $p_{l_v} = \frac{\exp(w_{l_v}(t))}{\exp(w_{l_v}(t))+1}$;
5:         b. $x_{l_v}(t) = 0$, Del($l_v$, $t \mod T$) otherwise;
6:     **else**
7:         $x_l(t) = 0$;
8:         Del($l$, $t \mod T$);
9: **for** each link $l_v \notin \mathcal{M}(t)$ **do**
10:     $x_{l_v}(t) = x_{l_v}(t-T)$;
11: 2. In the data slot, if $x_{l_v}(t) = 1$, include $l_v$ in $\mathcal{S}(t)$.
```

Note that we only need to maintain vectors $\mathbf{a}, \mathbf{e}, \mathbf{h}$ for T time-slots, e.g., $\mathbf{a}(0), \mathbf{a}(1), \cdots, \mathbf{a}(T-1)$ for $\mathbf{a}$. Now we only need to use our aforementioned procedures $AddCheck(l_v, t \mod T)$ and $Del(l_v, t \mod T)$ to update the state of $\mathbf{a}(t \mod T)$, $\mathbf{e}(t \mod T)$ and $\mathbf{h}(t \mod T)$ and check the feasibility constraint (3).

Proposition 3: Given $\epsilon > 0$, if we set the virtual link weight as

$$w_{l_v}(t) = \max\left\{ f_l(Q_l(t)), \frac{\epsilon}{2|\mathcal{E}_v|} f_l(Q_{\max}(t)) \right\}. \quad (9)$$

where $f_l(Q_l) = \log\log(Q_l + e)$. Then D-FlexCSMA is throughput-optimal for any finite delay time T.

We use the similar approach in [11] with minor modifications due to multiple RF chains and multiple link rates.

With D-FlexCSMA, we are able to de-correlate the schedules over time to reduce the delay. We will run numerical simulations in Section VI to evaluate the delay performance and compare the results with other scheduling algorithms. This is an initial attempt to improve the delay performance of FlexCSMA framework, it is still interesting to see how to effectively expedite the process of accumulating or depreciating link rates when the maximum number of RF chains on one FlexRadio node $m_{\max}$ is not small.

VI. Numerical Simulations

In this section, we use simulations to 1) verify the theoretical results of our proposed scheduling algorithms and 2) compare the performance of different scheduling algorithms, including FlexCSMA, D-FlexCSMA and MIMO-only CSMA, which directly applies traditional Q-CSMA framework to select non-interfering links and point-to-point MIMO is enforced.

A. A 5-Node FlexRadio Network

We first study a 5-node network as shown in Fig. 3 with 5 FlexRadio nodes (circles) A, B, C, D and E. We assume there are 6 RF chains on node C, 5 RF chains on node E, and the remaining nodes all have 3 RF chains. Furthermore, the interference range is assumed to be the same as the transmission range, and the interference relationship is indicated by a dotted line between nodes.

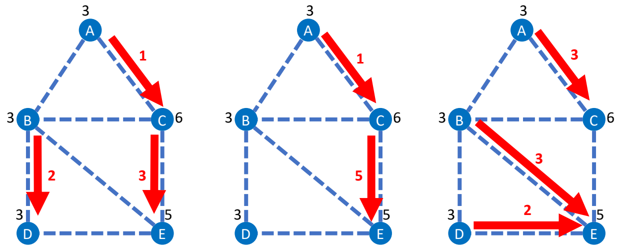

Fig. 3. A 5-node FlexRadio network with three maximal schedules

Recall that a maximal schedule is a feasible schedule such that adding any rate to any link will violate our feasibility constraint (3). We consider three maximal schedules $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ in Fig. 3. We use a red line and a number to denote an active data link and its associated link rate. Here, we do not specify the indices of the virtual links in each maximal schedule, since only the count matters.

Take $\mathcal{S}_1$ as an example, node C is using 1 RF chain to receive data from node A, 3 RF chains to transmit data to node E and the last 2 RF chains to withstand the interference streams from node B. At the same time, node E uses 3 RF chain to receive data from node C and 2 RF chains to withstand interference from node B. Node C and E have used up all their RF chains, hence, adding any more rate to any link will compromise the data reception at node C and E. In fact, this is a topology that has intense interference relationships, in which interference cancellation is expected to be quite useful in improving system throughput.

We set the arrival rate vector to be a convex combination of the link rates from maximal schedules $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ scaled by a utilization factor ρ, namely:

$$\lambda = \rho \sum_{i=1}^{3} \nu_i r^{\mathcal{S}_i}. \tag{10}$$

where $\nu = \{0.3, 0.3, 0.4\}$. If we choose the utilization factor $\rho < 1$, arrival vector λ stays inside the capacity region Λ. Any throughput-optimal scheduling algorithm should be able to stabilize λ for any positive number $\rho < 1$.

In our simulation, we use Bernoulli arrival process and each simulation runs for 10^7 time-slots. For activation probability p_l, we choose virtual link weight $w_{l_v}(t) = \log(0.1Q_l(t))$. In addition, we choose delay time $T = 200$ in D-FlexCSMA. The queue-length performance against different utilization vector ρ is shown in Fig. 4.

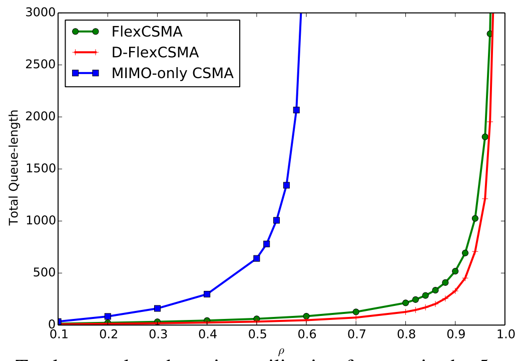

Fig. 4. Total queue-length against utilization factor ρ in the 5-node network

We have the following observations:

- Both FlexCSMA and D-FlexCSMA stabilize the queueing system for any $\rho < 1$, maximal schedules are usually achieved within 20 iterations. Compared to FlexCSMA, D-FlexCSMA saves nearly 35% of the total queue-length and packet delay.
- MIMO-only CSMA cannot stabilize the system for any $\rho > 0.6$, which indicates that the capacity region of FlexRadio is enlarged by more than 60% for this specific topology.

B. A 9-Node Ring Network

A 9-node ring network is shown in Fig. 5, where each circle stands for a FlexRadio node with 6 RF chains. Each node could be interfered by its two neighbors, i.e., one on each side. Considering two maximal schedules, $\mathcal{S}_1$ activates all

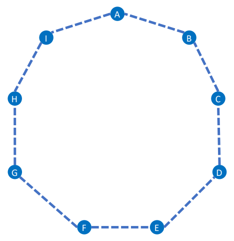

Fig. 5. A 9-node ring network with 6 RF chains on each node

clockwise links with rate 2, $\mathcal{S}_2$ activates all counter-clockwise links with rate 2. In both cases, each node is in Rx mode, and uses 2 RF chains to receive, transmit data and withstand interference streams. As in (10), we can define the arrival rate vector λ similarly with $\mathcal{S}_1$ and $\mathcal{S}_2$. It is easy to check that, compared with the MIMO-only transmission scheme, the additional flexibility of FlexRadio does not increase the capacity region of this specific network. We can easily apply the MIMO-only CSMA to stabilize any traffic pattern with $\rho < 1$. Now, the question remains: *Is it still worthwhile to use FlexRadio in this case?*

To answer this question, we run simulations to evaluate queue-length and delay performance. The results are shown in Fig. 6 and Table. I.

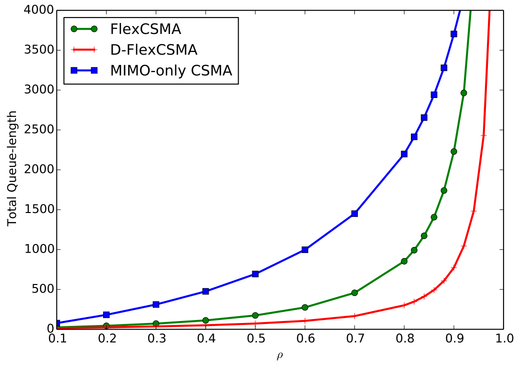

Fig. 6. Total queue-length against ρ in the 9-node ring network

TABLE I
DELAY COMPARISON BETWEEN DIFFERENT ALGORITHMS

ρ	0.3	0.5	0.7	0.9
MIMO-only CSMA	57.60	77.07	115.12	228.56
FlexCSMA	13.23	19.32	36.32	137.61
D-FlexCSMA	6.67	8.01	13.17	47.74

Although the FlexCSMA framework is more complicated than the MIMO-only CSMA, FlexCSMA framework is still able to converge very quickly and offers much better delay performance. The better delay performance is attributed to the flexibility of RF chain resource allocation, which is fully leveraged to meet different flow demands and provides faster response time in the case of bursty arrivals.

VII. Conclusion

FlexRadio, which allows flexible RF chain resource allocation, introduces a new feasibility constraint in wireless networks, which makes it challenging to design a high performance joint antenna allocation and link scheduling algorithm. In this paper, we use a novel model to characterize the feasibility constraint from the perspective of RF chain resource allocation. Based on this model, we propose a FlexCSMA scheduling framework that dynamically allocates and releases RF chain resources based on traffic rates and queueing dynamics in a distributed way. In addition, we prove that FlexCSMA achieves throughput-optimality and carry out numerical simulation to validate our results.

Acknowledgment

This work is funded in part by NSF grants CNS-1514260, CNS-1547306, CNS-1518829, and CNS-1254032, ONR grants N00014-17-1-2412, N00014-17-1-2417.

References

[1] Cisco visual networking index: Forecast and methodology, 2015-2020. http://www.cisco.com.

[2] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge University Press, 2005.

[3] C. B. Peel, Q. H. Spencer, A. L. Swindlehurst, and M. Haardt, "An introduction to the multi-user MIMO downlink," *IEEE communications Magazine*, vol. 61, 2004.

[4] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single channel, full duplex wireless communication," in *Proc. of Mobicom*. ACM, 2010, pp. 1–12.

[5] B. Chen, V. Yenamandra, and K. Srinivasan, "Flexradio: Fully flexible radios and networks." in *Proc. of NSDI*, 2015, pp. 205–218.

[6] Y. Yang, B. Chen, K. Srinivasan, and N. B. Shroff, "Characterizing the achievable throughput in wireless networks with two active RF chains," in *Proc. of IEEE INFOCOM*, 2014, pp. 262–270.

[7] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, 2006.

[8] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 3, pp. 960–972, 2010.

[9] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 3, pp. 825–836, 2012.

[10] P.-K. Huang and X. Lin, "Improving the delay performance of CSMA algorithms: A virtual multi-channel approach," in *Proc. of IEEE INFOCOM*. IEEE, 2013, pp. 2598–2606.

[11] J. Kwak, C.-H. Lee, D. Y. Eun, J. Kwak, C.-H. Lee *et al.*, "A high-order markov-chain-based scheduling algorithm for low delay in CSMA networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 4, pp. 2278–2290, 2016.

[12] Y. Yang and N. B. Shroff, "Scheduling in wireless networks with full-duplex cut-through transmission," in *Proc. of IEEE INFOCOM*, 2015, pp. 2164–2172.

[13] D. Qian, D. Zheng, J. Zhang, N. B. Shroff, and C. Joo, "Distributed CSMA algorithms for link scheduling in multihop MIMO networks under SINR model," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 3, pp. 746–759, 2013.

[14] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 4, pp. 1132–1145, 2009.