Abstract—In this paper, we aim to develop scheduling policies to maximize the stability region of a wireless network under the assumption that mutual information accumulation is implemented at the physical layer. This enhanced physical layer capability enables the system to accumulate information even when the link between two nodes is not good and a packet cannot be decoded within a slot. The result is an expansion of the stability region of the system. The accumulation process does not satisfy the i.i.d assumption that underlies many previous analysis in this area. Therefore it also brings new challenges to the problem. We propose two dynamic scheduling algorithms to overcome this difficulty. One performs scheduling every $T$ slot, which inevitably increases average delay in the system, but approaches the boundary of the stability region. The second constructs a virtual system with the same stability region. Through controlling the virtual queues in the constructed system, we avoid the non-i.i.d difficulty and attain the stability region. We derive performance bounds under both algorithms and compare them through simulation results.

I. INTRODUCTION

Cooperative communication has attracted much attention in the past few years. In wireless communications and networks, physical layer cooperation at nodes are exploited to increase the information theoretical capacity region, improve the energy efficiency, combat fading and interference, and reduce the probability of loss of connectivity.

Cooperative communication can be realized in various ways. In our work, the cooperation is carried out through mutual information accumulation (MIA) at nodes in a wireless network. Specifically, nodes can choose to overhear transmissions originated from other nodes. Although the overhead information is noisy, nodes retain whatever they hear and accumulate independent samples of the information until they can successfully determine what the message is.

Most of the current routing papers in the literature are based on energy accumulation. The difference between energy accumulation and mutual information accumulation is explained in [1]. Mutual information accumulation can be realized through the use of rateless (or fountain) codes [1]–[3].

While some research work focuses on optimizing resource allocation in networks with MIA [1], [4], in our work, under a stochastic setting, our objective is to characterize the maximum stability region when MIA is exploited, and design a joint routing and scheduling algorithm to achieve this region.


Similar to [5]–[7], we assume our system operates at packet-level, and each active transmitter transmits a packet in each slot. Different from the probabilistic channel model discussed in these references, we consider the scenario where channel states vary slowly, so that link conditions can be assumed static during a long transmission period. Under this scenario, for a weak link with link rate below the transmission rate, a packet cannot get through it in any single slot even with repetitive transmissions in multiple slots. Thus, the corresponding receiver can never become a possible forwarder under schemes discussed in [5]–[7]. However, when rateless codes are used at the transmitters, although the corresponding receiver of the weak link cannot decode the packet within a slot, it can store that corrupted packet and accumulate more information bits. Eventually, a packet can get through a weak link after a number of transmissions with information accumulation in multiple slots. Thus, the long, weak links can still be utilized in a slow changing network environment. Compared with opportunistic routing schemes, MIA provides a more reliable throughput over weak links, and it doesn’t require the feedback information to determine the forwarder at each slot, thus it reduces the overhead needed. Recently, [9] has done some parallel work on exploring the backpressure routing in wireless networks use MIA. In contrast to the static channel model discussed in our work, [9] investigates the problem under a fading channel model, where only the mean channel state information is available.
Compared to the networks in [10] and many other papers where varying rate can be achieved within a slot through adaptive encoding-decoding scheme, the MIA scheme doesn’t require the encoding-decoding to be changed in every slot, however, on average, it can achieve the same rate by repetitive transmission and information bits accumulation. Therefore, the system is more practical to implement without sacrificing any throughput. On the other hand, the information bit accumulation process also brings new challenges to the design of the optimal routing and scheduling algorithm.

The contribution of our work is three-fold.

- We characterize the capacity region of a wireless network with mutual information accumulation capability. Compared with networks where information accumulation is not allowed, the system is able to exploit weak links and an expanded capacity region can be achieved.
- We propose two dynamic routing and scheduling policies to achieve the capacity region. The policies require simple coordination and limited overhead.
- The techniques we develop to cope with non-i.i.d nature of the MIA process might have broader impacts on queueing problems in other areas.

II. SYSTEM MODEL

A. The Basic System Setting

We consider a time-slotted system with slots normalized to integral units $t \in \{0, 1, 2, \ldots\}$. There are $N$ network nodes, and links are labeled according to ordered node pairs $(i, j)$ for $i, j \in \{1, \ldots, N\}$. Data arrives randomly to the network in packetized units, where $A_i(t)$ is the number of packets that exogenously arrive at network node $i$ during slot $t$. Arrivals are assumed to be independent and identically distributed (i.i.d.) over timeslots, and we let $\lambda_i = E[A_i(t)]$ represent the arrival rate of packets into source node $i$ (in units of packets/slot). We assume $A_i(t) \leq A_{max}$ for all $i, t$. Similarly, we define $\mu_{max}$ as the maximum number of packets a node can successfully decode in any slot, which is equivalent to the maximum number of nodes that can transmit to a single node simultaneously.

For the brevity of analysis, we assume that all packets are from the same commodity, i.e., they have the same destination node, denoted as $des$. Our algorithm and analysis can be easily extended to scenarios where multi-commodity flows are allowed in the system.

We assume the channel fading states between any pair of nodes are stationary, and the active transmitters transmit with the same power level. Thus, a fixed reliable communication rate over an active link can be achieved in each slot. We expect that the algorithms developed in this paper can be slightly modified to accommodate more general fading processes.

We assume that at most one packet can be transmitted from any given node during a single time slot, i.e., $r_{ij} \leq 1$ packet/slot. We define the links with rate at least one packet per slot to be strong links; the rest of the links we name weak links. For weak links, we assume their rates are lower bounded by some constant value $r_{min}$, $0 < r_{min} < 1$. Define the set of neighbors of node $i$, $N(i)$, as the set of nodes with $r_{ij} > 0$, $j \in N(i)$. We assume the size of $N(i)$, denoted as $|N(i)|$, is upper bounded by a positive integer $d$.

We assume the system must operate under some constraints designed to reduce interference among transmitters. Under the interference model, in any timeslot, only a subset of nodes are allowed to transmit simultaneously. We assume that transmissions from nodes active at the same time are interference free from each other. We denote the set of feasible activation pattern as $S$, and an activation pattern $s \in S$ is represented by the set of active nodes. With a little abuse of the notation, we interchangeably use $s$ to represent an activation pattern and the set of active nodes of the pattern. This interference model can accommodate networks with orthogonal channels, restricted TDMA, etc.

B. Mutual Information Accumulation at Physical Layer

We assume MIA is adopted at physical layer. Specifically, we assume that if a weak link with $r_{ij} < 1$ is activated, rateless codes are deployed at its corresponding transmitter. Therefore, when a packet transmitted over a weak link cannot be successfully decoded during one slot, instead of discarding the undecodable message, the corresponding receiver stores that partial information, and accumulates more bits when the same packet is retransmitted. A packet can be successfully decoded when the accumulated mutual information exceeds the packet size. The assumption $r_{ij} > r_{min}$ implies that for any active link, it takes at most some finite number of time slots, $\lceil 1/r_{min} \rceil$, to decode a packet.

In order to simplify the analysis, we assume that $1/r_{ij}$ is an integer. The reason for this choice will become clear when our algorithm is proposed in Section III. If $r_{ij}$ does not satisfy this assumption, we round it down to $1/(1/r_{ij})$ to enforce it.

The challenge brought by mutual information accumulation can be illustrated through the following simple example. Suppose that a node $i$ has already accumulated half of the bits in packet 1 and half of the bits in packet 2. Since neither of the packets can be decoded though, none of these bits can be transferred to next hop, even the total number of bits at node $i$ is equal to that of a full packet. This implies that we should handle the undecoded bits in a different manner. We also observe that, if node $i$ never accumulates enough bits for packets 1 or 2, then these packets can never be decoded and will be stuck at node $i$. Coordination among the nodes is required to keep the buffers from overflowing. Moreover, unlike the opportunistic model studied in [5]–[7], given that weak link is active, whether or not a successful transmission will occur in that slot is not an i.i.d random variable but rather a deterministic function of the number of already accumulated bits of that packet at the receiving node.

Therefore, in the following sections, we define two different types of queues. One is the traditional full packet queue which stores fully decoded packets at each node. The other type of queue is a partial packet queue. It represents the fraction of accumulated information bits from a particular packet. The
specific definition of the queues and their evolution depends on the scheduling policy, and will be clarified in Section IV, and Section V, respectively.

We assume each node has infinite buffer space to store fully decoded packets and partial packets. Overflow therefore does not occur.

C. Reduced Policy Space

The policy space of the network we consider can be much larger than that of network without MIA. Because of the broadcasting nature of wireless communication, potentially, multiple receivers can accumulate the information of the same packet in a slot, and a receiver can collect information bits of a packet from multiple nodes. However, keeping track of the locations of different copies of the same packet requires a lot of overhead. Second, allowing multiple receivers keep the same packet actually introduces more traffic into the network, therefore, stabilizing the network requires sophisticated control strategy. Finally, accumulating information bits of a packet from multiple nodes makes the decoding options of a packet increase exponentially; thus characterizing the network capacity region becomes intractable. Therefore, we make the following assumptions:

A1. For each packet in the system, at any given time, only one node is allowed to keep a full copy of it.

A2. In addition to the node with the full copy, another node is chosen as the potential forwarder for the packet. Only the potential forwarder is allowed to accumulate the information about the packet.

Restricting ourselves to this policy space potentially sacrifices part of the stability region that can be achieved under any general policy. However, as we will see in the following, these assumptions enable us to characterize explicitly the maximum stability region with the given policy space. Compared with systems that operate without the capability to accumulate mutual information, our system is able to exploit weak links even when one-slot decoding is not possible. Thus the stability region is greatly enlarged.

If a node directly contributes to the successful decoding of a packet at another node, then, this node is denoted as a parent for that packet at a given time. Assumptions A1-A2 guarantee that for any packet in the network, there is only one parent at any given time. We also note that, if we relax assumptions A1-A2, and make the following assumption instead

A3. Every packet existing in the network has a single parent at any given time, i.e., the accumulated information required to decode a packet at a node is from a single node.

then, the maximum stability region under A1-A2 and that under A3 are equivalent. Assumption A3 means that we don’t allow a node to accumulate information from different nodes (different copies of the same packet) to decode a certain packet. However, there may exist multiple copies of a packet in the network.

III. NETWORK CAPACITY WITH MUTUAL INFORMATION ACCUMULATION

In this section we characterize the optimal throughput region under all possible routing and scheduling algorithms that conform to the network structure specified in Sec. II-A and Assumption A3.

At the beginning of each slot, a certain subset $s$ of nodes is selected to transmit in this slot. Any node $i \in s$ can transmit any packet that it has received and successfully decoded in the past. Note that because packets must be decoded prior to transmission, partial packets cannot be delivered. For a node $j \in N(i)$, if it decodes the packet before, it can simply ignore the transmission; otherwise, it listens the transmission and aim to decode it at the end of that slot. Receiving nodes connected with strong links can decode the packet in one slot. Nodes connected with weak links cannot decode the packet in one slot. They need to listen to the same node transmitting for the same packet over a number of slots in order to successfully decode the packet. A packet is successfully delivered to the destination when the first copy of the packet is decoded by the destination. These assumptions allow for any possible routing and scheduling policy satisfying Assumption A3.

Let $\lambda$ represent the input rate vector to the system, where $\lambda_i$ is the input rate entering node $i$. Define $Y_i(t)$ as the number of these packets that have been successfully delivered to the destination over $[0, t)$. According to the definition of network stability [7], a policy is defined as stable if $\lim_{t \to \infty} Y_i(t) = \lambda_i$.

The maximum stability region or network layer capacity region $\Lambda$ of a wireless network with MIA is defined as the closure of all $\lambda$ that can be stabilized by the network according to some policy with the structure described above.

**Theorem 1** For a network with given link rates $\{r_{ij}\}$ and a feasible activation pattern set $S$, the network capacity region $\Lambda$ under assumptions A3 consists of all rate vectors $\{\lambda_n\}$ for which there exists flow variables $\{\mu_{ij}\}$ together with probabilities $\pi_s$ for all possible activation patterns $s \in S$ such that

$$\begin{align*}
\mu_{ij} &\geq 0, \quad \mu_{\text{des},i} = 0, \quad \mu_{ii} = 0, \quad \forall i, j \\
\sum_i \mu_{ii} + \lambda_i &\leq \sum_j \mu_{ij}, \quad \forall i \neq \text{des} \\
\mu_{ij} &\leq \sum_{s \in S} \pi_s \theta_{ij}(s) r_{ij}, \quad \forall i, j \\
\sum_{s \in S} \pi_s &\leq 1
\end{align*}$$

where the probabilities $\theta_{ij}(s)$ satisfies

$$\theta_{ij}(s) = 0 \text{ if } i \notin s, \quad \sum_j \theta_{ij}(s) = 1, \forall i$$

The necessity of this theorem is proved following the same approach in [7] and can be found in [11]. The sufficiency part is proved in Section IV by constructing a stabilizing policy for any rate vector $\lambda$ that is in the interior of capacity region.
The capacity region is essentially similar to the capacity theorem of [7], [12]. The relations in (1) represent non-negativity and flow efficiency constraints for conservation constraints, and those in (2) represent flow conservation constraints, and those in (3) represent link constraints for each link \((i, j)\). The variable \(\theta_{ij}(s)\) can be interpreted as the probability that the transmission over link \((i, j)\) eventually contributes to the delivery of a packet at the destination, given that the system operates in pattern \(s\). In other words, link \((i, j)\) is on the routing path for this packet from its origin node to the destination node.

This theorem implies that the network stability region under Assumption A3 can be defined in terms of an optimization over the class of all stationary policies that use only single-copy routing. Thus, for any rate vector \(\lambda \in \Lambda\), there exists a stationary algorithm that can support that input rate vector by single-copy routing all data to the destination.

The \(\Lambda\) defined above are in the same form as when a “fluid model” is considered. In other words, the extra decoding constraint imposed by MIA does not sacrifice any part of the stability region. We can simply ignore the packetization effect when we search for the maximum stability region.

Solving for the parameters \(\pi_\ast\) and \(\theta_{ij}(s)\) required to satisfy the constraints requires a complete knowledge about the arrival rates \(\{\lambda_i\}_i\), which cannot be accurately measured or estimated in real networks. On the other hand, even when \(\Lambda\) is given, solving the equations directly can still be quite difficult. In the following, we overcome this difficulty with online algorithms which stabilize any \(\lambda\) arbitrarily close to the boundary of \(\Lambda\), with a possibly increased average delay in the system.

IV. T-SLOT DYNAMIC CONTROL ALGORITHM

In the following, we construct a policy that fits Assumptions A1-A2. Although more restrictive than Assumption A3, we will see that the stronger assumptions do not compromise stability performance, i.e., reduce the stability region. To construct a dynamic policy that stabilizes the system anywhere in the interior of \(\Lambda\) specified in Theorem 1, we first define our decision variables and queues as follows.

We assume that each packet entering the system is labeled with a unique index \(k\). At time \(t\), \(0 \leq k \leq \sum_{r=1}^{t} A_i(t)\). Let \(\{\beta_{ij}(k)(t)\}\) represent the binary control action of the system at time \(t\). Specifically, \(\beta_{ij}(k)(t) = 1\) means that at time \(t\), node \(i\) transmits packet \(k\) to node \(j\). We restrict the possible actions so that in each slot each node transmits at most one packet, and at most one node is chosen as the forwarder for packet \(k\). Therefore, the \(\beta_{ij}(k)(t)\) should satisfy the following constraints:

\[
\sum_{j,k} \beta_{ij}(k)(t) \leq 1, \quad \sum_{i,j} \beta_{ij}(k)(t) \leq 1, \forall i, j, k. \tag{6}
\]

Because of the MIA property, even if packet \(k\) is transmitted over link \((i, j)\) in slot \(t\), it doesn’t necessarily mean that packet can be decoded at node \(j\) at the end of slot \(t\). In particular, under the fixed link rate assumption, decoding cannot occur over weak links in one timeslot. We let \(f_{ij}(k)(t)\) be an indicator function where \(f_{ij}(k)(t) = 1\) indicates that the packet \(k\) has been successfully delivered from node \(i\) to node \(j\) in slot \(t\). The indicator function is a function of the current control action and partial queue status at the beginning of slot \(t\). Apparently, \(f_{ij}(k)(t) = 1\) implies that \(\beta_{ij}(k)(t) = 1\).

As discussed in Section II-B, we define two types of queues at each node. We use \(Q_i(t)\) to denote the length of node \(i\)’s queue of fully received packets at time \(t\), and use \(P_{i}(t)\) to represent the total fraction of packet \(k\) accumulated by node \(i\) up to time \(t\). The sum-length of partial queues at node \(i\) storing partial packets can be represented as \(P_i(t) = \sum_k P_{i}(k)(t)\). The fraction of packet \(k\), \(P_{i}(k)(t)\), can be cleared either when packet \(k\) is successfully decoded and enters the full packet queue \(Q_i\), or when the system controller asks node \(i\) to drop packet \(k\).

Then, according to our Assumptions A1-A2, the queue lengths evolve according to

\[
Q_i(t+1) = \left( Q_i(t) - \sum_{j,k} \beta_{ij}(k)(t)f_{ij}(k)(t) \right)^+ + \sum_{l,k} \beta_{il}(k)(t)f_{il}(k)(t) + A_i(t) \tag{7}
\]

\[
P_{i}(k)(t+1) = P_{i}(k)(t) + \sum_{l} \beta_{il}(k)(t)r_{il}(k)(t) - \sum_{l,m \neq i} P_{i}(k)(t)\beta_{im}(k)(t) \tag{8}
\]

where

\[
r_{il}(k)(t) = \begin{cases} r_{il} & P_{i}(k)(t) + r_{il} \leq 1 \\ 1 - P_{i}(k)(t) & P_{i}(k)(t) + r_{il} > 1 \end{cases} \tag{9}
\]

Under the assumption that \(1/r_{ij}\) is an integer for every \((i, j)\), we have \(r_{ii}(k)(t) = r_{ii}\).

Since we only allow there to be a single forwarder for any given packet at any time, if \(\beta_{im}(k)(t) = 1\), any nodes other than node \(m\) which have accumulated partial information of packet \(k\) must drop that partial packet \(k\). This effect results in the last negative term in (8). The first negative term corresponds to successful decoding of packet \(k\), after which it is removed and enters \(Q_i\).

A. The Algorithm

Our algorithm updates every \(T\) timeslots. The “optimal” update parameter \(T\) depends on the input rates and link capacities. We will analyze the effect of \(T\) on the stability region and average backlog performance. As we will see, when \(T\) is large enough, the following algorithm can stabilize any rate vector inside the capacity region \(\Lambda\).

1) Check single-link backpressure. At the beginning of an updating slot, i.e., when \(t = 0, T, 2T, \ldots\), node \(i\) checks its neighbors and computes the differential backlog weights

\[
W_{ij}(t) = [Q_i(t) - Q_j(t)]^+ r_{ij}, \quad j \in \mathcal{N}(i). \tag{10}
\]

2) Select forwarder. Choose the potential forwarder for the packets in \(Q_i\) as the node \(j\) with the maximum weight
Denote this node as \( j^*(i) = \arg \max_j W_{ij}(t) \).

3) **Choose activation pattern.** Define the optimal activation pattern \( s^* \) as the pattern \( s \in \mathcal{S} \) that maximizes \( \sum_{i \in s^*} W_{ij}^*(i) \). Any node \( i \in s^* \) transmits packets starting from the head of \( Q_i(t) \) to \( j^*(i) \). The pairing of transmitter \( i \in s^* \) and receiver \( j^*(i) \) is continued for \( T \) consecutive timeslots.

4) **Clear partial queues.** Release all the accumulated bits in the partial queue \( P_i(t) \), \( \forall i \), at the end of \( t = T - 1, 2T - 1, \ldots \).

We clear all of the partial queues in the system every \( T \) slots (in Step 4) for the simplicity of analysis. It is likely not be the best approach to handle the partial queues. Intuitively, the performance may be improved if we only release the partial queues when a selected forwarder for a packet is not the same as the previous one.

**Theorem 2**  The algorithm stabilizes any rate vector satisfying \( \lambda + \epsilon(T) \in \Lambda \), where \( \epsilon(T) \) is a vector with minimum entry \( \epsilon > 1/T \). The average expected queue backlog in the system is upper bounded by \( \frac{N T^2 (\mu_{\text{max}} + A_{\text{max}}) + N T^2}{2(\epsilon T - 1)} \).

The proof of this theorem can be found in [11]. The proof is based on the fact that the \( T \)-slot algorithm minimizes the \( T \)-slot Lyapunov drift, which is shown to be negative when \( \sum_i Q_i(t) \) is sufficiently large.

The constructed algorithm proves the sufficiency of Theorem 1. The intuition behind the algorithm is to utilize the weak links consecutively over a long window, such that the portion of dropped partial packet is kept to a very low level. Therefore, we don’t waste too much rates of the weak links. The algorithm approaches the boundary of the capacity region in the order of \( O(1/T) \).

We note that for some special values of \( T \), the network can still be stabilized even when \( T \leq 1/\epsilon \). For example, when \( T \) is chosen as \( \prod_{(i,j)} \alpha_{ij} = 1 \), then, under any possible activation pattern \( s \in \mathcal{S} \), all partial packets are decoded at the end of the \( T \)-slot window. This implies that the policy can stabilize any \( \lambda + \epsilon \in \Lambda \). For small networks, such \( T \) might be easily computed and be small; for large networks with many weak links, such value may still be quite large.

**V. Virtual Queue Based Algorithm**

In this section, we develop a second algorithm that achieves the maximum stability region without setting a large \( T \), thus it attains a better delay performance. As we have seen in Theorem 1, the delay is caused by the long time window of planning and infrequent update of the control action. Therefore, in order to obtain a better delay performance, intuitively, we need to update our policy more frequently. This requires us to design more sophisticated mechanism to handle the partial packet queues and non-i.i.d decoding process over weak links.

Our solution is to construct a network that contains “virtual” queues, which handle the partial packets and decoding process at weak links. The resulting network has the same maximum stability region as the original network. By stabilizing the constructed network, the original network is also stabilized.

Specifically, in order to handle the partial packet queue in a simple and effective way, we introduce buffers over weak links. We assume there is a buffer at the transmitter side for each weak link. Then, if a node wants to send a packet over a weak link, that packet is pushed into the buffer. The buffer keeps the packet until it is successfully decoded at the corresponding receiver. The intuition behind those buffers is that, since we don’t want to lose much effective rate over weak links caused by dropping partial packet, once the system picks a forwarding node for a particular packet, the system will never change this decision.

In order to overcome the non-i.i.d decoding process over weak links, we introduce a second buffer at the receiver side of each weak link. By controlling queue lengths, we ensure there is always enough accumulated information to attain successful decoding in this buffer. Thus decoding decisions can be made in an i.i.d fashion.

**A. The Virtual Queue Vector**

We divide \( Q_i(t) \) into two parts. The first stores the packets that have not yet been transmitted in any previous slots and is denoted as \( U_i(t) \). The second stores packets partially transmitted over some links before but have not yet been decoded, denoted as \( V_i(t) \). Since each packet in the second part is associated with some link, in order to prevent any “loss of transmission”, i.e., inefficiency, we require these packets to be transmitted over the same link until they are decoded. We use \( V_{ij}^{(k)}(t) \) to denote the information of packet \( k \) required to be transmitted over link \((i,j)\), and \( P_{ij}^{(k)}(t) \) to denote the accumulated information of packet \( k \) at node \( j \). We define \( V_{ij}(t) = \sum_k V_{ij}^{(k)}(t) \), and \( P_{ij}(t) = \sum_k P_{ij}^{(k)}(t) \). Note that \( P_{ij}(t) \) is different from \( P(t) \) defined in Section IV, since the latter is associated with node \( j \) and the former is associated with link \((i,j)\).

We follow the definition of control actions, where \( \{\beta_{ij}^{(k)}(t)\} \) represent the control action of the system at time \( t \). Depending on whether packet \( k \) belongs to \( U_i(t) \) or \( V_i(t) \), we also divide the decision actions into two types, denoted as \( \beta_{ij}^{(1)} \) and \( \beta_{ij}^{(2)} \), respectively. In the following algorithm, we only make control decisions for the packets at the head of corresponding queues. Therefore we can drop the superscript \( (k) \) without any worry of confusion. When \( \beta_{ij}^{(1)}(t) = 1 \), the transmitter pushes a new packet from \( U_i(t) \) into the tail of \( V_{ij} \) at the beginning of slot \( t \). This implies that the system assigns node \( j \) to be the next forwarder for that packet. Once the packet is pushed into \( V_{ij} \), we transmit the packet that is at the head of \( V_{ij} \) to node \( j \), thus, an amount of \( r_{ij} \) information can be accumulated at the tail of \( P_{ij}(t) \), and the length of \( V_{ij} \) is reduced by \( r_{ij} \). This mechanism ensures that the packets in the virtual buffer is transmitted and decoded in a FIFO fashion. When \( \beta_{ij}^{(2)}(t) = 1 \), without pushing a new packet into the buffer, we retransmit a packet at the head of \( V_{ij}(t) \). We let

\[
\beta_{ij}(t) = \beta_{ij}^{(1)}(t) + \beta_{ij}^{(2)}(t).
\]
We require that
\[ \sum_j \beta_{ij}(t) \leq 1, \quad \forall i, t. \] (11)

In addition, we define \( f_{ij}(t) \in \{0, 1\} \) as binary decoding control actions. \( f_{ij}(t) = 1 \) indicates that receiver \( j \) has accumulated enough information to decode the packet at the head of \( P_{ij}(t) \) and move it out of \( P_{ij}(t) \) and into \( U_{ij}(t) \). We impose the following constraint
\[ f_{ij}(t) \leq P_{ij}(t), \] (12)
which indicates that \( f_{ij}(t) = 1 \) only when \( P_{ij}(t) \geq 1 \), i.e., receiver \( j \) has accumulated enough information for a packet.

Then, according to constraints (11) and (12), the queue lengths evolve according to
\[ U_{ij}(t+1) = \left( U_{ij}(t) - \sum_j \beta_{ij}(t) \right)^+ + \sum_i f_{ij}(t) + A_{ij}(t) \] (13)
\[ V_{ij}(t+1) \leq (V_{ij}(t) + \beta_{ij}(t)(1 - r_{ij}) - \beta_{ij}(t)r_{ij})^+ \]
\[ = (V_{ij}(t) + \beta_{ij}(t) - \beta_{ij}(t)r_{ij})^+ \] (14)
\[ P_{ij}(t+1) \leq P_{ij}(t) + \beta_{ij}(t)r_{ij} - f_{ij}(t) \] (15)

The inequalities in (14) and (15) come from the fact that \( \beta_{ij}(t) \) can be applied to a dummy packet when a queue is empty. When the corresponding queue is not empty, the inequality becomes an equality.

Associated with each virtual queue, we define a virtual node, as depicted in Fig. 1. For the link from node \( i \) to node \( j \), we associate one virtual node with \( V_{ij} \) and a second with \( U_{ij} \). The virtual node associated with \( V_{ij} \) is denoted as \( v_{ij} \), while the virtual node associated with \( U_{ij} \) is denoted as \( u_{ij} \). We have decomposed the weak link \((i, j)\) into three links: \((i, v_{ij}), (v_{ij}, u_{ij}), (u_{ij}, j)\), with link rates \( r_{ij}, 1, 1 \), respectively. The virtual nodes and corresponding link rates for link \((j, i)\) can be defined in a symmetric way.

Define \( L(U(t), V(t), P(t)) = \sum_i U_{ij}^2(t) + \sum_{(i,j)} V_{ij}^2(t) + \sum_{(i,j)} (P_{ij}(t) - \eta)^2 \) where \( \eta \) is a parameter used to control the length of \( P_{ij}(t) \). Define \( \Delta(t) \) as the one-slot sample path Lyapunov drift:
\[ \Delta(t) := L(U(t+1), V(t+1), P(t+1)) - L(U(t), V(t), P(t)) \]

**Lemma 1** Under constraints (11) and (12), the sample path Lyapunov drift satisfies
\[ \Delta(t) \leq \alpha_2 + 2 \sum_i U_i(t)A_i(t) - 2 \sum_{i,j} (U_{ij}(t) - V_{ij}(t))\beta_{ij}(t) \]
\[ - 2 \sum_{i,j} (V_{ij}(t) - P_{ij}(t))r_{ij}\beta_{ij}(t) \]
\[ - 2 \sum_{i,j} (P_{ij}(t) - \eta - U_{ij}(t))f_{ij}(t) \] (16)
where \( \alpha_2 = N(d + A_{max})^2 + 2N + Nd \).

The proof of this lemma is provided in Appendix A.

**B. The Algorithm**

In contrast to the algorithm of section IV, this algorithm updates every timeslot. The purpose of the algorithm is to minimize the right hand side of (16) given the current \( U, V, P \).

1) **Find per-link backpressure.** At the beginning of a timeslot, node \( i \) checks its neighbors and computes the differential backlogs. We compute the weight for the link between node \( i \) and the first virtual node, and, separately, the weight for the link between the two virtual nodes. Specially, the weight for control action \( \beta_{ij}^1 \) is computed as
\[ W_{ij}^1(t) = (U_{ij}(t) - V_{ij}(t) + (V_{ij}(t) - P_{ij}(t))r_{ij})^+ \]
and the weight for control action \( \beta_{ij}^2 \) is computed as
\[ W_{ij}^2(t) = (V_{ij}(t) - P_{ij}(t))r_{ij} \]
The weight for the link \((i, j)\) is \( W_{ij}(t) = \max\{W_{ij}^1(t), W_{ij}^2(t)\} \).

2) **Select forwarder.** Choose the potential forwarder of the current slot for node \( i \) with the maximum weight \( W_{ij}(t) \) and denote it as \( j^*(i) \).

3) **Choose activation pattern.** Define the optimal activation pattern \( s^* \) as the pattern that maximizes \( \sum_{i \in s} W_{ij}(i) \).

4) **Transmit packets.** For each \( i \in s^* \), let node \( i \) transmit to node \( j^*(i) \). If \( W_{ij}(t) = W_{ij}^1(t) \), node \( i \) pushes a new packet from \( U_i \) into \( V_{ij} \) and transmits the packet from the head of \( V_{ij} \); otherwise, node \( i \) resends the packet at the head of \( V_{ij} \).

5) **Decide on decoding actions.** For each link \((i, j)\), we choose \( f_{ij}(t) \in \{0, 1\} \) to maximize
\[ [P_{ij}(t) - U_{ij}(t) - \eta]f_{ij}(t) \] (17)
while satisfying \( f_{ij}(t) \leq P_{ij}(t) \).

**Lemma 2** Under the above virtual queue based algorithm, (a) if \( P_{ij}(t) < \eta \) for some weak link \((i, j)\) and slot \( t \), then \( f_{ij}(t) = 0 \). (b) If \( P_{ij}(to) \geq \eta - 1 \), then, under the proposed algorithm, \( P_{ij}(t) \geq \eta - 1 \) for every \( t \geq t_0 \).

**Proof:** In order to maximize \( (P_{ij}(t) - \eta - U_{ij}(t))f_{ij}(t) \), \( f_{ij}(t) = 1 \) only when \( P_{ij}(t) - \eta - U_{ij}(t) > 0 \). Therefore, if \( P_{ij}(t) < \eta \), \( f_{ij}(t) \) must equal zero, which proves (a).

Now suppose that \( P_{ij}(t) \geq \eta - 1 \) for some slot \( t \). We show that it also holds for \( t+1 \). If \( P_{ij}(t) \geq \eta \), then it can decrease by at most one packet on a single slot, so that \( P_{ij}(t+1) \geq P_{ij}(t) - f_{ij}(t) \geq \eta - 1 \). If \( P_{ij}(t) = \eta - 1 \), we must have \( f_{ij}(t) = 0 \) and the queue cannot decrease in slot \( t \), and we again have \( P_{ij}(t+1) \geq \eta - 1 \).
With Lemma 2, we can see that when setting \( \eta = 1 \), under the proposed algorithm, if \( P_{ij}^c(t) < 1 \) for some weak link \((i,j)\) and slot \( t \), then \( f_{ij}^c(t) = 0 \). \( f_{ij}^c(t) \) only possibly equals one when \( P_{ij}^c(t) \geq 1 \). Thus, constraint (12) is satisfied automatically for every slot under the proposed algorithm.

**Theorem 3** For a network with given link rates \( \{r_{ij}\} \) and a feasible activation pattern set \( S \), the network capacity region \( \Lambda' \) for the constructed network consists of all rate matrices \( \{\lambda_n\} \) for which there exist flow variables \( \{\mu_{ij}^v\}, v = 1,2,3 \) together with probabilities \( \pi_s \) for all possible activation pattern \( s \in S \) such that

\[
\begin{align*}
\mu_{ij}^v & \geq 0, \quad \mu_{i,des,i}^v = 0, \quad \mu_{ii}^v = 0, \quad \forall i,j,v \\
\sum_i \mu_{i}^v + \lambda_i & \leq \sum_j \mu_{ij}^v, \quad \forall i \neq des \\
\mu_{ij}^1 & \leq \mu_{ij}^2, \quad \mu_{ij}^2 \leq \mu_{ij}^3, \quad \forall i,j \\
\mu_{ij}^3 & \leq \sum_{s \in S} \pi_s \theta_{ij}(s)r_{ij}, \quad \forall i,j \\
\sum_{s \in S} \pi_s & \leq 1
\end{align*}
\]

(18) (19) (20) (21) (22)

where the probabilities \( \theta_{ij}(s) \) satisfies

\[
\theta_{ij}(s) = 0 \text{ if } i \notin s, \quad \sum_j \theta_{ij}(s) = 1, \forall i
\]

(23)

**Proof:** The necessary part can be proved in the same way for Theorem 1. The sufficiency will be proved through constructing an algorithm to stabilize all rate vectors satisfying the constraints.

In this constructed virtual network, \( \mu_{ij}^1, \mu_{ij}^2, \mu_{ij}^3 \) can be interpreted as the flow over links \((i,v_{ij}), (v_{ij}, p_{ij}), (p_{ij}, j)\), respectively. The constraints (18) represent non-negativity and flow efficiency constraints for conservation constraints. The constraints in (19), (20) represent flow conservation constraints, where the exogenous arrival flow rates for nodes \( v_{ij}, p_{ij} \) are zero. The constraints in (21) represent the physical link constraint for virtual link \((v_{ij}, p_{ij})\), which equals the link constraint for the real link \((i,j)\) in the original system. Note that there is no explicit link constraints for \((i,v_{ij})\) and \((p_{ij}, j)\), since the transfer of packets over these links happen at the same node, and there is no physical link constraints for them.

**Lemma 3** The network capacity region for the virtual network \( \Lambda' \) defined in Theorem 3 is equal to that for the original system \( \Lambda \) defined in Theorem 1.

**Proof:** First, we show that if \( \Lambda \in \Lambda' \), then it must lie in \( \Lambda \) as well. This can be shown directly by letting \( \mu_{ij} = \mu_{ij}^3 \). Thus, we have \( \mu_{ij}^1 \leq \mu_{ij} \leq \mu_{ij}^3 \). Plugging into (19), we have (2), i.e., if \( \Lambda \) satisfies the constraints in (18)-(23), it must satisfy (1)-(5) as well. Thus \( \Lambda \in \Lambda \).

The other direction can be shown in the following way: we prove that for any \( \Lambda + \epsilon \in \Lambda \), \( \Lambda + \frac{\epsilon}{d+1} \in \Lambda' \), where \( d \) is the maximum degree of the network.

Since \( \Lambda + \epsilon \in \Lambda \), we have

\[
\sum_i \mu_{ii} + \lambda_i + \epsilon \leq \sum_j \mu_{ij}, \quad \forall i \neq des.
\]

(24)

By letting \( \mu_{ij}^2 = \mu_{ij} \), we have that (21)-(23) satisfied. At the same time, we let \( \mu_{ij} + \epsilon_1 = \mu_{ij}^3 - \epsilon_1 = \mu_{ij}^3, \) and plug them into (24), which gives

\[
\sum_i (\mu_{i}^3 - \epsilon_1) + \lambda_i + \epsilon \leq \sum_j (\mu_{ij}^3 + \epsilon_1), \quad \forall i \neq des.
\]

(25)

Therefore,

\[
\sum_i \mu_{i}^3 + \lambda_i + \epsilon - 2\epsilon_1 \leq \sum_j \mu_{ij}, \quad \forall i \neq des.
\]

(26)

By letting \( \epsilon - 2\epsilon_1 = \epsilon_1 \), we have

\[
\sum_i \mu_{i}^3 + \lambda_i + \epsilon_1 \leq \sum_j \mu_{ij}, \quad \forall i \neq des
\]

(27)

\[
\mu_{ij}^1 + \epsilon_1 = \mu_{ij}^2, \quad \mu_{ij}^2 + \epsilon_1 = \mu_{ij}^3.
\]

(28)

Thus, we have \( \Lambda + \epsilon_1 \in \Lambda' \). As \( \epsilon \rightarrow 0 \), \( \epsilon_1 \) approaches zero as well. Thus, \( \Lambda = \Lambda' \).

**Theorem 4** By letting \( \eta = 1 \), the proposed algorithm stabilizes any rate vector satisfying \( \Lambda + \epsilon \in \Lambda \). The average expected queue backlog in the system is upper bounded by \( (2d+1)(N(d + A_{max})^2 + 2N + 3Nd)/\epsilon \).

The proof of the theorem is given in Appendix B.

The algorithm updates every slot. This avoids the delay caused by infrequent policy updating in the \( T \)-slot algorithm. On the other hand, we introduce virtual queues in the system. Since the differential backlog in Step 1) of the algorithm is not the physical differential backlog between nodes \((i,j)\) in the real system, the inaccuracy of the queue length information potentially increases the average backlog in the system. This is reflected by the \( 2d+1 \) factor in the upper bound. For arrival rate vector which is close to the boundary of network capacity region, the \( T \)-slot algorithm can only stabilize the system with a large \( T \), thus the virtual-queue based algorithm attains a better delay performance in heavy traffic regime.

The algorithm stabilizes the network without any traffic statistics. Compared to the \( T \)-slot algorithm which only approaches the maximum stability region with a carefully selected \( T \), this is a big advantage.

**VI. Simulation Results**

In this section we present detailed simulation results. These results exemplify the basic properties of the algorithms.

We consider a 4-node wireless network shown in Fig. 2 where the links with nonzero rates are shown. We assume new packets destined for node 4 arrive at node 1 and node 2 according to independent Bernoulli processes with rate \( \lambda_1 \) and \( \lambda_2 \), respectively. Node 3 does not have incoming packets, acting purely as a relay. We assume that the system does not have any activation constraints so that all nodes can transmit simultaneously without causing interference to each other.
The maximum stability region $\Lambda$, shown in Fig. 3, is the union of rate pairs $(\lambda_1, \lambda_2)$ defined according to Theorem 1. If MIA is not allowed, the corresponding maximum stability region is the triangle inside $\Lambda$. This is because when weak links are not utilized, the only route from node 1 to node 4 passes through node 2, thus the sum of arrival rates from node 1 and 2 cannot exceed link rate 1. When MIA is exploited, the weak link from node 1 to node 4 with rate $1/9$ can be utilized, thus an expanded stability region can be achieved.

We first compare the performance of the $T$-slot algorithm for different values of $T$. For each $T$, we conduct the simulation for arrival rates $\lambda_1 = \lambda_2 = \lambda$ ranging from 0 to 0.55. The resulting average backlog curve is shown in Fig. 4. As we can see from the figure, when $T = 1$, the weak links are not utilized at all, so the algorithm can only stabilize the arrival rates up to $\lambda = 1/2$, which is point $A$ in Fig. 3. When $T = 9$, which is the reciprocal of the link rate $1/9$, the algorithm can stabilize the arrival rates up to $\lambda = 9/17$, corresponding to point $B$ in Fig. 3. In this case, all of the partial packets transferred over weak link (1, 4) are eventually decoded, and the weak link is fully utilized. This is a special scenario since the value of $T$ perfectly matches the rate of weak link. For larger networks that consist of many weak links, selecting $T$ to match all of the weak links may be prohibitive, since such value can be very large. For more general values of $T$, the weak link is partially utilized, therefore, the maximum $\lambda$ the algorithm can stabilizes is some value between 1/2 and 9/17. In general, a larger $T$ stabilizes larger arrival rates, and results in an increased average backlog in the system. This is illustrated by curves with $T = 15, 60$ in Fig. 3.

We then provide the performance simulation for the virtual queue based algorithm. As we can see in Fig. 3, the system can be stabilized up to $\lambda = 9/17$. Compared with the $T$-slot algorithm, the virtual queue based algorithm attains a much better delay performance for large value of $\lambda$, i.e., when the system is in heavy traffic regime. It dominates even the curve with $T = 9$ at high rates. For small values of $\lambda$, the virtual queue based algorithm does not show much advantage in terms of delay. This is because the algorithm requires the virtual queues to build up certain lengths in order to push packets through the weak links. The virtual queue based algorithm has a relatively universal delay performance for $\lambda \in [0, 1/2]$, while under the $T$-slot algorithm, the average backlog increases significantly when $\lambda$ increases.

VII. CONCLUSIONS

In this paper, we analyzed the optimal routing and scheduling policy when MIA is exploited at the physical layer in a wireless network. We first characterized the maximum stability region, which is shown to surpass the network capacity region when MIA is not allowed. Two scheduling policies are proposed to cope with the non-i.i.d decoding process introduced by MIA. Both can achieve the maximum stability region but the latter has significantly reduced delay in heavy traffic regime. We also compared the performance under these two policies analytically and numerically.

ACKNOWLEDGEMENT

The authors would like to thank Andreas F. Molisch, Michael Neely, and Hao Feng for many stimulating and helpful conversations.

APPENDIX A

THE PROOF OF LEMMA 1

Based on (13)-(15), we have

$$U_i^2(t+1) \leq U_i^2(t) - 2U_i(t)\left(\sum_j \beta_{ij}(t) - \sum_l f_{il}(t) - A_i(t)\right) + \left(\sum_j \beta_{ij}(t)\right)^2 + \left(\sum_l f_{il}(t) + A_i(t)\right)^2$$

(29)

$$V_{ij}^2(t+1) \leq V_{ij}^2(t) + 2V_{ij}(t)\left(\beta_{ij}(t) - \beta_{ij}(t)r_{ij}\right) + \left(\beta_{ij}(t) - \beta_{ij}(t)r_{ij}\right)^2$$

(30)

$$P_{ij}^2(t+1) \leq P_{ij}^2(t) + 2P_{ij}(t)(\beta_{ij}(t)r_{ij}(t) - f_{ij}(t)) + (\beta_{ij}(t)r_{ij}(t) - f_{ij}(t))^2$$

(31)

$$P_{ij}(t+1) \geq P_{ij}(t) - f_{ij}(t)$$

(32)
Thus,
\[
\Delta(t) \leq \sum_i -2U_i(t) \left( \sum_j \beta_{ij}(t) - \sum_i f_i(t) - A_i(t) \right) \\
+ \sum_{(i,j)} 2V_{ij}(t) \left( \beta_{ij}(t) - \beta_{ij}(t)r_{ij} \right) \\
+ \sum_{(i,j)} 2P_{ij}(t) (\beta_{ij}(t)r_{ij}(t) - f_i(t)) + 2\eta f_i(t) + C
\]
where
\[
C = \sum_i \left( \sum_j \beta_{ij}(t) \right)^2 + \sum_i \left( \sum_j f_i(t) + A_i(t) \right)^2 \\
+ \sum_{(i,j)} (\beta_{ij}(t) - \beta_{ij}(t)r_{ij})^2 + \sum_{(i,j)} (\beta_{ij}(t)r_{ij}(t) - f_i(t))^2
\]
Because of constraints (11)-(12), we have
\[
C \leq N(d + A_{max})^2 + 2N + Nd =: \alpha_2.
\]
Combining items with respect to link \((i, j)\), we have (16).

**APPENDIX B**

**THE PROOF OF THEOREM 4**

**Corollary 1** A rate rate vector \(\lambda + \epsilon\) is in the capacity region \(\Lambda'\) if and only if there exists a stationary (possibly randomized) algorithm that chooses control decisions (independent of current queue backlog) subject to constraints (11), to yield

\[
E\{\beta_{ij}\} r_{ij} = E\{\beta_{ij}^{\ast}\} + \epsilon
\]

\[
E\{f_{ij}\} = E\{\beta_{ij}\} r_{ij} + \epsilon
\]

\[
E\left\{ \sum_j f_{ij} - \sum_i \beta_{ij}^{\ast} - \lambda_i \right\} \geq \epsilon \quad \forall i \neq des
\]  

The result is an immediate consequence of Theorem 3. The intuition is to let \(E\{\beta_{ij}\} r_{ij} = \mu_{ij}^2\), \(E\{\beta_{ij}^{\ast}\} = \mu_{ij}^1\) and \(E\{f_{ij}\} = \mu_{ij}^3\).

**Lemma 4** Under the virtual queue based algorithm,

\[
\sum_{i,j} \left\{ (U_i(t) - V_{ij}(t))\beta_{ij}(t) + (V_{ij}(t) - P_{ij}(t))r_{ij}\beta_{ij}(t) \right\} \\
+ (P_{ij}(t) - \eta - U_j(t))f_{ij}(t) \right\} \\
\geq \sum_{i,j} \left\{ (U_i(t) - V_{ij}(t))\beta_{ij}(t) + (V_{ij}(t) - P_{ij}(t))r_{ij}\beta_{ij}(t) \right\} \\
+ (P_{ij}(t) - \eta - U_j(t))f_{ij}(t)
\]

for any other policy \(\{\hat{\beta}_{ij}, \hat{\beta}_{ij}^{\ast}, \hat{f}_{ij}\}\) satisfying (11).

Based on Lemma 1 and Lemma 4, we have
\[
E\{\Delta(t)|U(t), V(t), P(t)\} \\
\leq -2 \sum_{i,j} E\left\{ (U_i(t) - V_{ij}(t))\beta_{ij}(t) + (V_{ij}(t) - P_{ij}(t))r_{ij}\beta_{ij} \right\} \\
+ (P_{ij}(t) - \eta - U_j(t))\hat{f}_{ij}(t) \left| U(t), T(t), P(t) \right\} \\
+ 2 \sum_i U_i(t)\lambda_i + E[C]
\]

\[
\leq E\left\{ \sum_i -2U_i(t) \left( \sum_j \beta_{ij}(t) - \sum_i \hat{f}_{ij}(t) - \lambda_i(t) \right) \right\} \\
+ E\left\{ \sum_{(i,j)} 2V_{ij}(t) \left( \beta_{ij}(t) - \hat{\beta}_{ij}(t)r_{ij} \right) \right\} \\
+ E\left\{ \sum_{(i,j)} 2P_{ij}(t) (\hat{\beta}_{ij}(t)r_{ij}(t) - \hat{f}_{ij}(t)) + 2\eta \hat{f}_{ij}(t) \right\} + \alpha_2
\]

Since for \(\lambda + \epsilon \in \Lambda\), we have \(\lambda + \epsilon/(2d + 1) \in \Lambda'\), thus when \(\eta = 1\),

\[
E\{\Delta(t)|U(t), V(t), P(t)\} \\
\leq -2 \left( \sum_i U_i(t) + \sum_{(i,j)} V_{ij}(t) + \sum_{(i,j)} P_{ij}(t) \right) \frac{\epsilon}{2d + 1} \\
+ 2Nd + N(d + A_{max})^2 + 2N + Nd.
\]

Therefore, the system is stable, and the average backlog is upper bounded by

\[
\lim_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \left( \sum_i U_i(\tau) + \sum_{(i,j)} V_{ij}(\tau) + \sum_{(i,j)} P_{ij}(\tau) \right) \\
\leq \frac{(2d + 1)(N(d + A_{max})^2 + 2N + 3Nd)}{\epsilon}
\]

The proof is completed.

**REFERENCES**


