Queueing Systems manuscript No.
(will be inserted by the editor)

Scheduling with pairwise XORing of packets under statistical overhearing information and feedback

Georgios S. Paschos · Leonidas Georgiadis · Leandros Tassiulas

the date of receipt and acceptance should be inserted later

Abstract We study the problem of scheduling packets from several flows traversing a given node which can mix packets belonging to different flows. Practical wireless network coding solutions depend on knowledge of overhearing events which is obtained either by acknowledgments or statistically. In the latter case, the knowledge about each packet improves progressively with feedback from the transmissions. We propose a virtual network mechanism in order to characterize the throughput region of such a system for the case where we allow only pairwise XORing. We also provide the policy which achieves the stability region and compare it to simple heuristics. The derived policy is a modification of the standard backpressure policy, designed to take into account the fact that in the proposed virtual network the destination of a transmitted packet is known only probabilistically. We demonstrate simulation results according to which scheduling with statistical information can provide significant throughput benefits even for overhearing probabilities as small as 0.6.

1 Introduction

Network coding (NC) has emerged as one of the most promising techniques in the effort to improve network performance [3,16,9]; NC has been theoretically shown to achieve the multicast capacity of wireless networks [3]. The capacity boost due to NC has been shown even in simple scenarios with *wireless network coding*, where an encoded packet travels at most 1 hop [16,24]. In [16], the authors proposed

G. S. Paschos and L. Tassiulas are with University of Thessaly, E-mail: gpasxos@gmail.com, leandros@uth.gr · L. Georgiadis is with Aristotle University of Thessaloniki, E-mail: leonid@auth.gr
The authors are with CERTH-ITI as well. Their work was supported by European Commission FP7 STAMINA-265496 and Marie Curie CodeLance-285969 research projects.
A preliminary version of this work was published in [22] which mainly summarized the problem and the results focused on the network coding scenario. Additionally to providing proofs, the current extended version also contains several examples as well as a general version of the virtual network approach which can be used in other application scenarios as well. The complete version of the current paper is available online at the Queueing Systems Journal www.springer.com.

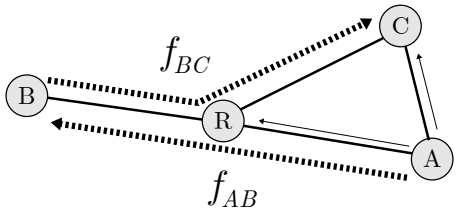

Fig. 1 Opportunistic listening; When Alice uploads her packet destined to Bob, Chloe overhears the transmission and stores the packet as a key. Later, the relay node may mix the flows f_{AB} and f_{BC} successfully. Bob decodes due to ownership and Chloe due to overhearing.

COPE, the first experimental evidence of throughput gain via wireless NC with XOR operations.

The strength of XOR operations lies in the simplicity of the decoding functionality. Given that $N-1$ out of N coded packets are known to a receiver, the application of an XOR operation will provide the N^{th} packet. We call these $N-1$ packets *keys* for this particular receiver. The theoretical gain of wireless NC in the simple Alice–Relay–Bob topology [16] is $\frac{4}{3}$, since instead of using four transmissions to serve the *symmetric flows* (two flows i, j are called symmetric if the source of i is the destination of j and vice versa), only three are sufficient. The maximum possible theoretical gain harvested in one network junction is 2 and it arises in the so-called infinite wheel topology, see [16]. Similar gains have been reported in the case of applying intersession NC at each node of a long path, see [5, 25], while the combination of local coding and intersession coding over long paths may potentially reach higher throughput gains as explained in [12]. In this paper, we focus on local coding; the coded packets are decoded one hop away from the encoding node.

The actual throughput gain in a practical scenario varies greatly with the topology, the transmission rates and the probability of an erroneous transmission due to fading or interference on the wireless link. Nevertheless, wireless network coding is deemed a promising technique for increasing the throughput of wireless links since the provided gains aggregate throughout the whole network and they can be implemented without affecting the operation of the rest of the protocol stack.

In order to increase the efficiency of the wireless NC scheme, COPE utilizes *opportunistic listening*, see Figure 1. This technique is a smart way of exploiting the broadcast medium; first devices need to store copies of all native packets that they overhear from the common medium. Second, routers need to be informed of the key packets that each neighboring receiver maintains, towards making encoding decisions. As a result, this extra functionality enables encoding of packets belonging to non-symmetric flows and thus extends the throughput benefits to wider topologies and flow scenarios. Obtaining this overhearing information at the decision point (relay node) is not a trivial task though. The first approach is to deal with this issue by *explicitly* acknowledging all overheard packets, a policy reported

by the authors of [16] to be sluggish and costly as the channel rate and number of neighbors increases.

As an alternative lightweight approach, [16] proposes obtaining statistical information about the overhearing events. The packet overhearing information is guessed by routers, by utilizing link quality advertisements through periodic probing. This is a feature automatically provided by SRCR routing protocol [1] that most of practical NC implementations are built on, e.g. [16,23,24,4]. Avoiding monitoring each overhearing event is experimentally shown to relieve router CPU of a heavy burden, [16]. This comes of course at a loss of throughput; the set of arrival rates stabilizable by the system with deterministic knowledge but not stabilizable with only statistical knowledge is called regret region (the scheduler regrets not having deterministic state information). In this paper, we are interested to quantify this loss.

In this paper, we focus on wireless NC and study the problem of scheduling packets and making encoding decisions jointly. Driven by the applicative nature of scheduling of XORs with statistical overhearing information, we make the following contribution:

- We formulate the problem of joint scheduling and encoding selection taking into account two distinct cases:
  1. the relay has deterministic information about all key owners,
  2. the relay only has statistical overhearing information and utilizes feedback from the transmissions.
- For the deterministic case:
  - We give the throughput optimal (stabilizing) policy for the case of F flows and XORing only pairs of packets. The policy is of backpressure type, and is derived based on the standard framework leading to such types of policies.
  - We discuss the extension to coding arbitrarily many packets.
- For the statistical case:
  - We propose a virtual network technique which categorizes the packets based on the belief of the scheduler about the key owners.
  - Using this virtual network, we characterize the throughput region of the system in the case of F flows and allowing for pairwise encoding (i.e. encoding up to two packets).
  - We provide the throughput optimal policy for this case. The policy is of backpressure type again, applied on the virtual network. There is an important difference from the classical framework, however, *the destination of a transmitted packet is random and uncontrollable*. We rigorously deal with this technical issue and determine the resulting backpressure policy.
- We perform simulations with different settings showing the throughput benefits obtained from scheduling with statistical information.

The rest of the paper is organized as follows. In Section 2 we discuss previous work. In Section 3 we describe in length the NC scheme and the system model we deal with. We also provide further motivation for our work. Next, in Section 4, we formulate the problem stating also some basic definitions. In Section 5 we deal with the problem in case of deterministic overhearing information and in Section 6 in case of statistical overhearing information and feedback. In Section 7 we provide simulations and we conclude our work in Section 8. Appendices I and II provide the mathematical proofs while Appendix III extends the applicability

of the algorithm to cases where the decisions involve transmissions with multiple packets. Also, Appendix IV, explains why coding more than pairs, i.e. for $|\mathcal{P}| > 2$, requires the development of further techniques.

2 Related Previous Work

In this Section, we discuss previous studies on wireless NC, and differentiate our work. Our work is based on the practical system of [16], named COPE. The authors of [16] implemented a wireless network coding scheme which allows both for deterministic and statistical overhearing knowledge. However, [16] does not consider the problem of scheduling and utilizes a simple heuristic mechanism for treating the statistical case, explained in detail in Section 7. In this work, we plan to extend the treatment of the statistical case and solve the problem of making encoding decisions and performing scheduling jointly.

Apart from [16], there exist other practical approaches that try to enable beyond-COPE practices in real systems using XOR operations. CLONE for example deals with lossy links and relaxes the constraint of immediate decoding. The authors show that even for the binary NC the problem of determining the optimal encoding sequence is $\#P$–complete, [23]. In this process, it is assumed that encoded packets may be stored at receivers. In ER, [24], the authors consider the problem of selecting the set of packets to code together towards minimizing the number of retransmissions and show that it is NP–hard. This result can be parallelized to the work in the context of index coding where we are given a number of information bits partially known by the receivers and we search for the transmission policy that minimizes the time to complete reception by all receivers, [7]. All these approaches assume that the scheduler/encoder (or the entity that makes the transmission decisions) has deterministic knowledge of which packets (keys) each destination has. Our work builds on top of the stochastic approach proposed in [16] and addresses the problem of making encoding decisions based solely on statistical information. There are several practical advantages of the latter, such as reduced overhead, reduced latency in decision making and smooth scheduler operation. More importantly, [16] advises that operation in high channel rates (6Mbps in that case) mandates the use of statistical information only.

The authors in [5] observe that NC should be carried out jointly with scheduling (otherwise NC may reduce throughput). Then, they propose a simple practical scheme, XOR-SYM which allows XOR coding of symmetric flows only and disregards opportunistic listening. Our work is in line with [5], with the difference that we focus on a single hop case that utilizes opportunistic listening. With symmetric flows as in [5], knowledge of overhearing events is not required and the scheduling problem is simpler since feedback does not affect the decisions.

There exist other works dealing with scheduling in network coding-enabled networks as well, see for example [8, 13, 14]. A common characteristic of those dynamic strategies with our work is that they do not require arrival rates as input. The on-line system is robust to dynamics because it reacts to present circumstances using the state of the queues. The difference in our work, lies on the fact that contrary to these works, we do not assume complete knowledge for the system. Instead, we assume that the scheduler possesses only statistical information about the overhearing events, i.e. which destination holds which keys.

Scheduling in systems with uncertainty and feedback is a topic of recent interest, [6,10]. In these previous works it is stressed that in the general case, such problems are well modeled by Markov decision processes which however are often intractable to solve. In our work, we make a connection between scheduling problems with feedback and the control theory of [11] which results in feasible algorithms for solving such problems optimally.

The problem of scheduling packets, when only partial network state information (NSI) is available, is studied in [28,15]. From prior work, it is known that delayed queue length information does not affect the throughput region, but delayed channel conditions for example can indeed diminish the throughput region. In [28], the authors assume that the channel state evolves as a Markov chain which leads to algorithms that optimally control the arising Markov decision process. The structure of the chain is used in the calculations of the expected backpressure metric. This work has some similarity with our work, in that it addresses the problem of routing and scheduling in an environment where there is uncertainty. While, however, in [28] the uncertainty regards the channel state and affects the number of successfully transmitted packets under a given control, in our case uncertainty regards the overhearing of packets; this latter uncertainty is reduced progressively through the knowledge acquired by the received feedback, and the objective is to use this knowledge efficiently. This motivates the introduction of virtual network which captures the knowledge the scheduler has about the state of each packet.

Virtual queues have been used earlier in the context of handling complications that may arise in network control, as well as addressing problems with performance constraints. In [14], the concept of virtual queues is utilized in order to handle multiple copies of multicast packets. Problems involving performance constraints, e.g., upper bounds on node power consumption, have been solved by introducing virtual queues in [21], [26]. The problem of sampling a random channel and scheduling packets has been studied jointly in [19], where time average metrics are optimized subject to stability and the satisfaction of time average penalty constraints. The author proposes a virtual queue mechanism for capturing time average constraints and a modified backpressure algorithm that provides the solution to the problem. The approach considered in the current paper is differentiated from the previous works in that it is based on the creation of a virtual network for capturing and utilizing the knowledge obtained through feedback. The virtual network has the property that the recipient of a transmitted message is (uncontrollably) random, a case which is not captured by the existing framework nor encountered, to our knowledge, in any work in the literature. Thus, apart from the main contribution, which is the mapping of the problem to the virtual network, a technical contribution that this paper offers is to modify the standard approach in order to solve this added complexity.

3 Network Coding scheme description

We assume the existence of 2–hop flows traversing a central node and we write $i \in \mathcal{F} = \{1, 2, \ldots, F\}$ such that $i : \sigma_i \to \mathrm{R} \to \delta_i$ where σ_i denotes the source node of flow i, δ_i denotes its destination node and R is the central node called the relay. The relay maintains a queue for each flow, call it n^i for flow i and let X^i be its

corresponding backlog. It also possesses a scheduler that makes packet encoding and transmission decisions.

In the **uplink phase**, the flow sources transmit native packets towards the relay, populating the corresponding queue backlog X^i there, see Figure 2. All destination nodes, constantly listen to the common medium and overhear some of the transmitted packets. Each node maintains an infinite buffer where it stores all overheard native packets (as explained in the introduction, we assume that encoded packets are not allowed to be stored). Also, each source node stores in its buffer the transmitted packets since these packets can also be used for encoding/decoding. As described above, the overhearing events take place during the uplink phase and this is the main reason for describing this phase in detail. In the current work we concentrate on the problem of scheduling in the downlink phase. Transmissions in the uplink are considered to take place independently of the downlink through, e.g., a time sharing mechanism.

We denote the packets of flow i arriving at the relay with P_k^i, where $k = 1, 2, \ldots$ orders the packets based on arrival time (we may omit the subscript k when we speak of an arbitrary packet). Each arriving packet P_k^i is associated with an *overhearing state vector* (we will use the term overhearing vector) $\mathbf{s}_k^i$ that characterizes the destinations that overheard this packet. Specifically, $\mathbf{s}_k^i$ is a binary vector taking values in $\mathcal{S} = \{0,1\}^F$ with $s_k^i(j) = 1$ indicating that node δ_j (the destination of flow j) has (by overhearing or ownership) the packet P_k^i in its buffer and $s_k^i(j) = 0$ indicating the complement. We assume that $\mathbf{s}_k^i$ is random and the randomness is stirred by channel fading, mobility or spatially differentiated collisions. Note here that if for some flows i, j we have $\delta_i = \delta_j$ then $s_k^\ell(j) = s_k^\ell(i), \forall \ell \in \mathcal{F} \setminus \{i, j\}$ since the receiving node is physically the same for these two flows. In such cases, the updates of the overhearing vector should occur simultaneously for the two flows (why this is important will become apparent later).

Next we define two possible ways of conveying overhearing information to the relay.

Deterministic information: each node announces the overhearing of each packet to the relay by sending specific control packets. In this scheme (used by COPE, CLONE, etc), the relay obtains deterministic information about key knowledge at the expense of increased complexity. That is, the relay scheduler is assumed to know deterministically $\mathbf{s}_k^i$ for all flows i and all packets P_k^i in the system.

Statistical information: the relay obtains statistical information about the overhearing events through a mechanism that operates independently from the scheduler and at a larger time-scale. This scheme is used by COPE for high channel rates and it is the focus of this paper. In this scheme, the relay does not know the overhearing vector $\mathbf{s}$. Instead it initially knows the probability of the event $s_k^i(j) = 1$ denoted as q_{ij} and collected in the overhearing probability matrix $\mathbf{Q}$. Utilizing feedback information, the relay may improve the knowledge about the state of a packet each time a decoding failure occurs. As will be seen below, this implies that the expected service rate obtained by serving a given queue is not constant, a fact that gives opportunities for performance improvements but complicates the scheduling decisions.

In the **downlink phase**, the relay may select a set of packets $\mathcal{P}$ chosen from different flows, encode them in a single packet (perform per bit XOR) and transmit the encoded packet at the minimum rate $\min_i\{r_i^{down}\}$ where the minimization is

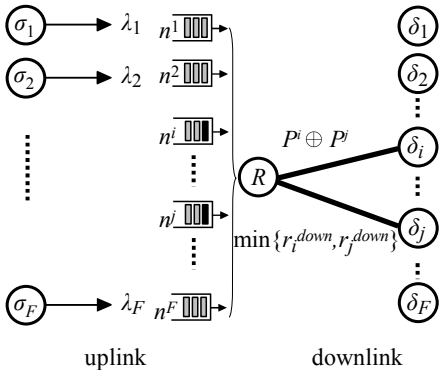

Fig. 2 Overview of our scheme; We study scheduling in the downlink but consider also the overhearing events that take place in the uplink.

over the set of flows to which the packets in $\mathcal{P}$ belong. In this paper we limit ourselves to the case where $|\mathcal{P}| \leq 2$. Thus in what follows, the available controls of the relay scheduler are either to transmit a native packet or to apply the XOR operator on two selected packets from different flows.

In the **feedback phase**, each intended destination δ_i attempts to decode the encoded packet and obtain the corresponding P_k^i, and in case of success returns an *acknowledgment message* (ACK). Once the ACK is received, the relay removes P_k^i from queue n^i which is now considered served. This is always true when the relay transmits a native packet. In case of an encoded packet however, we have four distinct cases which convey complete feedback information to the relay about the overhearing events of the transmitted packets. More specifically, say we transmit $P_{k_1}^i \oplus P_{k_2}^j$, we have the following cases:

1. Both packets are ACKed in which case they both leave the system.
2. Packet $P_{k_1}^i$ is ACKed and packet $P_{k_2}^j$ is not ACKed. In this case, $P_{k_1}^i$ leaves the system and $P_{k_2}^j$ stays in the system while the relay learns that δ_i owns $P_{k_2}^j$ as a key (i.e. learns that $s_{k_2}^j(i) = 1$) and it may use this information in the future.
3. Packet $P_{k_1}^i$ is not ACKed and packet $P_{k_2}^j$ is ACKed. By symmetry, $P_{k_1}^i$ stays in the system and the relay learns that δ_j owns $P_{k_1}^i$ as a key (i.e. learns that $s_{k_1}^i(j) = 1$) and $P_{k_2}^j$ leaves the system.
4. Both packets are not ACKed in which case they both stay in the system while the relay learns that δ_j does not own $P_{k_1}^i$ and δ_i does not own $P_{k_2}^j$ (i.e. learns that $s_{k_1}^i(j) = s_{k_2}^j(i) = 0$).

Note that the feedback informs the relay of the true state of the two particular packets $P_{k_1}^i$ and $P_{k_2}^j$ and not of any other packet belonging to the same flows. Thus, after an encoded packet transmission, the relay learns the element of the packet overhearing vector that corresponds to the pairing flow. However, if the packet is destined to a node which happens to be the destination node of another flow as well, the element that corresponds to this flow is learnt as well.

Here we summarize the assumptions of our model that affect the throughput region of a general multihop network:

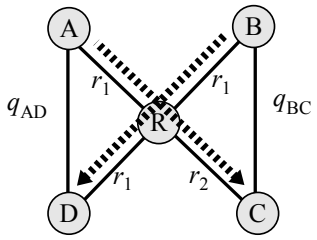

Fig. 3 An example network with two flows, A→R→C and B→R→D. D overhears A with probability q_{AD} and C overhears B with probability q_{BC}.

1. We concentrate on the problem of single hop downlink scheduling.
2. Use XORs for combining packets (instead of more general operations, e.g. linear combinations of packets, where each packet is considered as an element of a finite field).
3. Terminals are not allowed to forward encoded packets (local NC), i.e. they must decode immediately the packet and then encode it again if appropriate at the next hop.
4. Terminals are not allowed to store encoded packets for future use; they only store native packets.
5. Transmissions to intended recipients are errorless, however these transmissions may be overheard by other nodes with certain probability.

Assumptions 2,3 have been widely used in prior system development and contribute to making the network coding scheme easy to implement and maintain. We plan to relax assumptions 1,4,5 in future work.

Next we provide further motivation for our approach.

3.1 Encoding decision motivation

Consider the simple network of Figure 3 where two flows exist, A→R→C and B→ R →D. D overhears A with probability q_{AD} and C overhears B with probability q_{BC} and let $q_{\mathrm{AD}} = 1$, $q_{\mathrm{BC}} = q$, hence after transmission of an encoded packet $P_{AC} \oplus P_{BD}$, destination D always decodes packet P_{BD} while destination C may or may not decode packet P_{AC}. Consider the downlink rates $r_C^{down} \doteq r_2$ and $r_D^{down} \doteq r_1$ and the uplink rates $r_A^{up} = r_B^{up} \doteq r_1$, and assume equal arrival rates $\lambda_1 = \lambda_2 = \lambda$ [1]. Under strategy 1: "Transmit only native packets" the maximum λ achieved is $\frac{2r_1r_2}{3r_2+r_1}$ while under strategy 2 "Transmit the encoded packet $P_{AC} \oplus P_{BD}$ and retransmit as native all undecoded P_{AC} packets" it is $\frac{2r_1r_2\min\{r_1,r_2\}}{2r_2\min\{r_1,r_2\}+r_1r_2+(1-q)r_1\min\{r_1,r_2\}}$. By setting these two equal, the threshold q_{thr} is obtained, such that for $q \geq q_{thr}$ NC is beneficial. Solving, we get

$$q_{thr} = \begin{cases} 0 & r_1 \leq r_2 \\ \frac{r_1-r_2}{r_1} & r_1 > r_2, \end{cases}$$

[1] Here we have slightly changed the notation to make it more explanatory. Following the initial notation we should have $q_{\mathrm{AD}} \equiv q_{12}$, $q_{\mathrm{BC}} \equiv q_{21}$, $P_{\mathrm{AC}} \equiv P^1$, $P_{\mathrm{BD}} \equiv P^2$, $r_A^{up} \equiv r_1^{up}$, $r_B^{up} \equiv r_2^{up}$, $r_C^{down} \equiv r_2^{down}$, $r_D^{down} \equiv r_1^{down}$.

indicating that the strategy "encode as many packets as possible" used in [16] (where equal rates are assumed) is not optimal when $r_1 > r_2$. Also, a fixed threshold policy is bound to fail in a variable rate scenario. The phenomenon becomes non-trivial to visualize and solve if one considers several flows with some of them having common sources or destinations.

3.2 Scheduling with feedback motivation

To see the impact of feedback, consider the cross topology of Figure 3 with $q_{AD} + q_{BC} \geq 1$ and $q_{AD} < 1$, $q_{BC} < 1$ and $r_1 = r_2$. Assume the use of a reasonable scheduling policy which selects the packet $P_{AC} \oplus P_{BD}$ iff $q_{AD} + q_{BC} \geq 1$ and the transmission of a native packet otherwise. This corresponds to using NC when the expected decoding is at least one packet on the average. Now assume that we only have two packets to send, one for each flow, and it happened that these packets were both not overheard at uplink time. The scheduler, oblivious of this unhappy occasion, will encode the two packets (since $q_{AD} + q_{BC} \geq 1$) and transmit the coded packet. In this case no ACK will be received since it is impossible for both destinations to decode. A scheduler that disregards feedback information will keep on sending the encoded combination of the same packets resulting in a deadlock. Note, that we may let the above packets mix in the queues with other packets and thus reduce the probability of a deadlock. However, the throughput will decrease in any case. The reason this happens is because the scheduler actually has in its possession new information which is not taken into account in future scheduling decisions. Indeed, once no ACK messages were received, the scheduler can deduce the information that none of the two packets is overheard. Then it should append this information to the particular packets and treat them differently.

4 Problem Formulation

Consider the downlink of a network with F flows and F corresponding queues, similar to the one in Figure 2. We consider a slotted system where the time unit is the length of a slot. Slot $\tau = 1, 2...$ covers the time interval $[\tau - 1, \tau]$. During slot τ, $A_i(\tau)$ packets of flow i arrive at the corresponding queue at the relay node along with a randomly chosen overhearing vector $\mathbf{s}_k^i$ for the kth arriving packet. We assume that the process $A_i(\tau)$ consists of independent random variables with rate $\mathrm{E}\{A_i(\tau)\} = \lambda_i$.

At the beginning of each slot, the scheduler located at the relay chooses one control I from a set of controls $\mathcal{I}$. Each control consists of activating either a single queue, or a pair of queues. A control having only one queue activated corresponds to a native packet transmission. A control with two activated queues corresponds to the transmission of the XOR of two packets. Although the chosen control dictates which packets are transmitted, the service (successful transmission) of a packet in a queue is determined also by the overhearing vector $\mathbf{s}_k^i$ which is uncontrollable. We use the definitions of queue network stability from [11].

Definition 1 (Queue stability) The queue n^i of flow i is called stable if

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathrm{E}\Big\{ X^i(\tau) \Big\} < \infty.$$

Definition 2 (Network stability) The network is stable if all queues are stable.

A complete treatment of the problem would require to define stability to include the overhearing buffers as well. A simple way to remove this extra complexity is to have the relay send a "flush overheard packet queues" message whenever the relay queues become empty. This does not alter the analysis of stability of the queues at the relay. Thus, for the rest of the paper we will deal only with the relay queues.

We seek to find a policy π which selects an appropriate control at the beginning of each slot such that the downlink of the network is stabilized. The *stability region* of a policy π, Λ_π is the closure of the set of arrival rates for which the network is stable when π is in use. The *throughput region* Λ is the closure of the set of all arrival vectors $\boldsymbol{\lambda} = \{\lambda_i\}$ stabilizable by any policy. A control policy π^* is called *throughput optimal* if its stability region is Λ, see [11,27] for implications of these definitions.

5 Systems with deterministic overhearing information

In a system that operates with overhearing acknowledgments, such as COPE, CLONE and ER, packet decoding information is provided deterministically at the scheduler. Consequently, the feedback does not add information and thus scheduling in this case is straightforward.

Define the set of all possible overhearing vectors for flow i:

$$\mathcal{S}_i = \{\mathbf{s} \in \{0,1\}^F : s(i) = 0\}.$$

Next we assume that the relay creates 2^{F-1} queues for each flow i, each queue corresponding to a different overhearing vector $\mathbf{s} \in \mathcal{S}_i$. The set of queues is

$$\mathcal{N} = \{n^i_{\mathbf{s}}, i \in \mathcal{F}, \mathbf{s} \in \mathcal{S}_i\}.$$

Therefore, each queue corresponds to a different combination of destination nodes having overheard the packets backlogged there. Upon the arrival of a packet in the uplink, the relay checks the overhearing vector of the packet and places it in the appropriate queue.

For an illustrative example, say we have three flows $\mathcal{F} = \{1,2,3\}$. The relay constructs 12 queues, 4 for each flow, of the type $n^i_{\mathbf{s}}$, where for $i = 1,2,3$ and $\mathbf{s} = (s(1), s(2), s(3))$, $s(i) = 0$ and the rest of the components take values in $\{0,1\}$. Next a packet of flow 1 (say P^1_k) arrives at the relay along with the information that it was overheard from the destination of flow 2 but not from the destination of flow 3, i.e. the overhearing vector is $\mathbf{s}^1_k = (0,1,0)$. Then the relay will assign this packet to the queue $n^1_{(0,1,0)}$.

As a scheduling control the relay may choose to transmit a native packet from one of the available $F2^{F-1}$ queues or alternatively select two packets from two

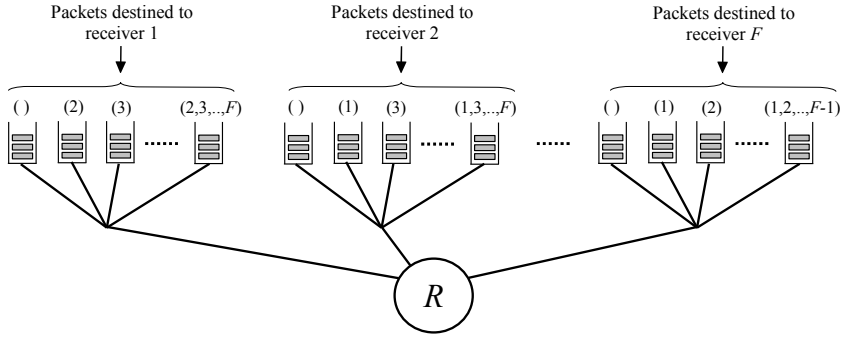

Fig. 4 Deterministic case: for each flow, 2^{F-1} queues keep packets with different overhearing vectors. The numbers in the parentheses show the flows the destination of which has correctly overheard the queued packets.

queues belonging to different flows and transmit the encoded combination. Formally, we define the controls as carefully chosen subsets of the queue set $\mathcal{N}$. For a singleton control, serving queue $n^i_{\mathbf{s}}$, that belongs to flow i and has overhearing vector $\mathbf{s} \in \mathcal{S}_i$, we write $I = \{n^i_{\mathbf{s}}\}$ for the control. The service rate will be $\hat{\mu}_{n^i_{\mathbf{s}}}(I) = r_i^{down}$ independently of the overhearing vector, and zero for all other queues. For pair controls mixing queue $n^i_{\mathbf{s}_1}$ of flow i with queue $n^j_{\mathbf{s}_2}$ of flow $j \neq i$ with $\mathbf{s}_1 \in \mathcal{S}_i$ and $\mathbf{s}_2 \in \mathcal{S}_j$ we write $I = \{n^i_{\mathbf{s}_1}, n^j_{\mathbf{s}_2}\}$ and we have

$$\hat{\mu}_{n^i_{\mathbf{s}_1}}(I) = \begin{cases} \min\{r_i^{down}, r_j^{down}\} & \text{if } s_2(i) = 1 \\ 0 & \text{if } s_2(i) = 0 \end{cases}$$

and similarly for $\hat{\mu}_{n^j_{\mathbf{s}_2}}(I)$ by exchanging i with j and s_2 with s_1. Also, the rates induced by this control on queues other than $n^i_{\mathbf{s}_1}$ and $n^j_{\mathbf{s}_2}$ are zero.

A throughput optimal algorithm in this case is a max-weight type algorithm that weights the queue backlogs with the service rates for each control from the set $\mathcal{I}$ and selects the one that maximizes the sum of this product. Denote with $X^i_{\mathbf{s}}$ the backlog of queue $n^i_{\mathbf{s}}$.

Algorithm 1: At each decision slot:

1. For each control $I = \{n^i_{\mathbf{s}_1}\}$ form the reward $C(I) = X^i_{\mathbf{s}_1}\hat{\mu}_{n^i_{\mathbf{s}_1}}(I)$.
2. For each control $I = \{n^i_{\mathbf{s}_1}, n^j_{\mathbf{s}_2}\}$ form the reward $C(I) = X^i_{\mathbf{s}_1}\hat{\mu}_{n^i_{\mathbf{s}_1}}(I) + X^j_{\mathbf{s}_2}\hat{\mu}_{n^j_{\mathbf{s}_2}}(I)$.
3. Then select $I^* = \arg\max_{I\in\mathcal{I}}\{C(I)\}$.

Proposition 1 *Algorithm 1 is throughput optimal for a system with deterministic overhearing information.*

The Proposition is a straightforward application of the standard framework for stochastic networks, see [18] and [11].

From this algorithm, it follows immediately that we can restrict pair controls to queues $n^i_{\mathbf{s}_1}, n^j_{\mathbf{s}_2}$ for which $s_1(j) = s_2(i) = 1$. To see this, assume that we have $s_2(i) = 0$, in which case $\hat{\mu}_{n^i_{\mathbf{s}_1}}(I) = 0$. Then the reward of the control $I = \{n^i_{\mathbf{s}_1}, n^j_{\mathbf{s}_2}\}$ is

$$C(I) = X^i_{\mathbf{s}_1}\hat{\mu}_{n^i_{\mathbf{s}_1}}(I) + X^j_{\mathbf{s}_2}\hat{\mu}_{n^j_{\mathbf{s}_2}}(I) = X^j_{\mathbf{s}_2}\hat{\mu}_{n^j_{\mathbf{s}_2}}(I)\,,$$

which is at most equal to the reward of control $I' = \{n^j_{\mathbf{s}_2}\}$ and therefore removing I does not affect the optimization in step 3 of Algorithm 1. For reasons of complexity, we can utilize this property and eliminate an important number of controls from the set.

In a node serving F flows, at each slot the scheduler must select the best weight out of $F2^{F-1}\left(1+2^{F-2}(F-1)\right)$ available controls. When the arrival rates are in the interior of the stability region of the system and not too close to the boundary, most of these queues are expected to be empty, and therefore the difficulty is simply to design a dynamic list of such size. Many of these controls can be eliminated since they are dominated by other controls (as explained above). Also, when the channel state changes slowly, there exist methods to simplify the search over the space of available controls, i.e. it is possible to update at each slot only the weights for the controls that have changed during the last slot. In [2] the interested reader may find CPU performance results of similar algorithms in real systems.

Extending the above algorithm to the case where we allow $|\mathcal{P}| > 2$ is fairly straightforward. The available controls are subsets of the set of queues $\mathcal{N}$ with the property that no two queues from the same flow are selected:

$$\mathcal{I} = \{I \subset \mathcal{N} : \forall n^i_{\mathbf{s}_1}, n^j_{\mathbf{s}_2} \in I \text{ with } n^i_{\mathbf{s}_1} \neq n^j_{\mathbf{s}_2} \text{ we have } i \neq j\}.$$

Also we need to extend the service rate function. First, let $\mathcal{F}_I \subseteq \mathcal{F}$ be the set of flows to which the queues in I belong. Also for each flow $i \in \mathcal{F}_I$ pick a vector from $\mathcal{S}_i$ and name it $\mathbf{s}_i$. For each queue $n^i_{\mathbf{s}_i}, i \in \mathcal{F}_I$ we get:

$$\hat{\mu}_{n^i_{\mathbf{s}_i}}(I) = \begin{cases} \min_{j\in\mathcal{F}_I}\{r_j^{down}\} \text{ if } s_j(i) = 1 \quad \forall j \in \mathcal{F}_I \setminus \{i\} \\ 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Conclusively, the case where the relay obtains deterministic knowledge about overhearing events can be handled by the given framework even when we allow for coding arbitrarily many packets together. Nevertheless, note that computationally the optimization problem in Algorithm I becomes much more complex. Thus, restricting to pairwise XORing might still represent an appealing practical solution.

6 Systems with statistical overhearing information

In a system where the scheduler possesses only statistical overhearing information, scheduling becomes highly non-trivial, mainly because of feedback. An encoded packet may not be successfully decoded because the corresponding destination may not have overheard a key packet. In such a case, the scheduler must utilize the feedback information if it is to make correct decisions. Moreover, the selection of a control affects the efficiency of future controls, a property that poses further complications in the analysis. To see this, consider an example of two packets from two different flows, such that one is correctly overheard from the one destination node and destined to the other (call it a good packet), while the other is not heard by any of the two destination nodes (call it a bad packet). In the first scenario, we transmit the XOR combination of the two, in which case the bad packet is correctly received while the good remains in the system and the feedback tells us that it is overheard by the other destination. In the second scenario, we simply

transmit natively the bad packet which gets received. If we compare the network state of the two scenarios, we see that the queues hold the same packets but the scheduler in the first case knows the overhearing state of the bad packet while in the second not. This extra knowledge can be utilized for increasing the efficiency of scheduling. Also, the expected reward calculated by the scheduler is different depending on past control decisions. Such a case is not allowed in the framework in [11]. Also, previous works [28,15] have studied cases of such dependence where the state of the link qualities evolves as a Markov process. Here, however, we deal with the complication of feedback from the overhearing events and a modification of the approach is required.

To reflect the feedback changes on our queue-based scheduling algorithm we propose a virtual network mechanism. The high-level idea is that the virtual nodes hold in their queues packets with a given overhearing state. This way, it is guaranteed that each packet is present in only one virtual node.

6.1 The virtual network mechanism

To illustrate the virtual network mechanism we start with the case of two flows ($F = 2$) and we allow the transmission of either native packets, or the XOR combination of the pair.

6.1.1 Virtual network for two flows

We will use the notation $i, \bar{i}$ with $\bar{1} \doteq 2$ and $\bar{2} \doteq 1$. Any particular packet P_k^i of flow i can be categorized by the scheduler into different states according to the estimation of the event $s_k^i(\bar{i}) = 1$: "the destination of flow $\bar{i}$ has P_k^i" as follows:
Unknown state (u): In this case the scheduler has only statistical information about $s_k^i(\bar{i})$, i.e it knows $\mathrm{P}\big(s_k^i(\bar{i}) = 1\big) \doteq q_{i\bar{i}}$, where we allow $0 \leq q_{i\bar{i}} \leq 1$.
Good state (g): In this case the scheduler knows that $s_k^i(\bar{i}) = 1$.
Bad state (b): In this case the scheduler knows that $s_k^i(\bar{i}) = 0$.

In order to group packets with the same properties together, we define for each flow i a directional subnetwork (subgraph) $\mathcal{G}_i = (\mathcal{N}_i, \mathcal{E}_i)$ having one node for each possible packet state and one for the destination. The destination node serves as a sink for all packets of the same flow and thus, once a packet reaches the destination node, it disappears from the virtual network. We write $\mathcal{N}_i = \{n_u^i, n_g^i, n_b^i, n_d^i\}$, $i = 1, 2$ for the virtual node set. In order to capture the possible state transitions, we define the virtual link set to consist of the ordered pairs $\mathcal{E}_i = \{e_{ud}^i, e_{gd}^i, e_{bd}^i, e_{ug}^i, e_{ub}^i, e_{gg}^i, e_{bb}^i\}$, with $e_{km}^i \doteq (n_k^i, n_m^i)$ and $i = 1, 2$. The virtual network will be the union of the two subnetworks for $i = 1, 2$, see Figure 5.

We associate each node of the virtual network with a queue (i.e. node-queue n_u^i holds packets of flow i being at unknown state). The packets of flow i enter the virtual network at the node n_u^i, they are routed inside the network and eventually leave the system when they reach node n_d^i -thus the destination node queue is always empty. At each time slot, the scheduler selects a control which corresponds to activating either a) one node from the virtual network or b) two nodes from two different subnetworks, excluding destination nodes. Once a control is taken, the first packet in each selected node (head-of-line, HOL, packet) is transmitted. In

I	w^1_{ud}	w^1_{ug}	w^1_{ub}	w^1_{gd}	w^1_{gg}	w^1_{bd}	w^1_{bb}
$\{n^1_u, n^2_u\}$	q_{21}	$(1-q_{21})q_{12}$	$(1-q_{21})(1-q_{12})$	0	0	0	0
$\{n^1_u, n^2_g\}$	1	0	0	0	0	0	0
$\{n^1_u, n^2_b\}$	0	q_{12}	$1-q_{12}$	0	0	0	0
$\{n^1_g, n^2_u\}$	0	0	0	q_{21}	$1-q_{21}$	0	0
$\{n^1_g, n^2_g\}$	0	0	0	1	0	0	0
$\{n^1_g, n^2_b\}$	0	0	0	0	1	0	0
$\{n^1_b, n^2_u\}$	0	0	0	0	0	q_{21}	$1-q_{21}$
$\{n^1_b, n^2_g\}$	0	0	0	0	0	1	0
$\{n^1_b, n^2_b\}$	0	0	0	0	0	0	1

Table 1 Transition weights in the subnetwork $\mathcal{G}_1$ for all pair controls.

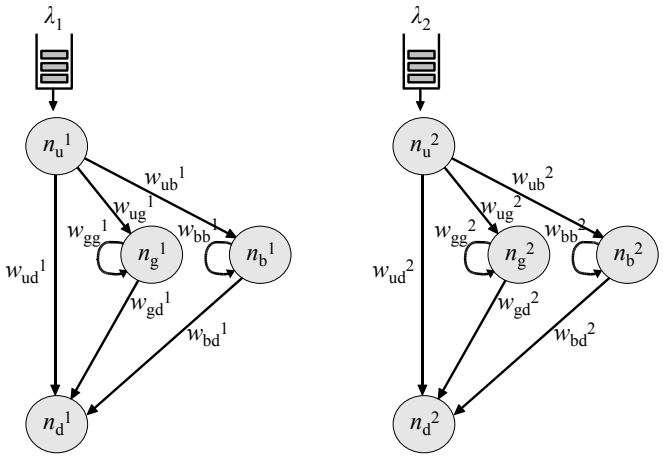

Fig. 5 The virtual network for the case of 2 flows consists of two subnetworks, one for each flow.

general, the recipient of a transmitted packet is determined randomly according to a probabilistic law that depends on the chosen control. While the scheduler knows the probability law, it does not know the actual recipient - unless the probability of transition to a particular recipient is 1. To reflect this, each link (k,m) of subnetwork $\mathcal{G}_i$ is associated with a probability weight $w^i_{(k,m)}(I)$ that depends on the control taken (sometimes and if there is no possibility for confusion we may omit superscript i for simplicity). For any given control I and any node k we have $\sum_{m\in\mathcal{N}^k} w_{(k,m)}(I) = 1$ where $\mathcal{N}^k$ is the set of neighbors of k.

Next we describe all possible transitions of a packet of flow 1 when the control $I = \{n^1_u, n^2_u\}$ is taken. For simplicity, for the remaining of this Section as well as Figure 5 and Table 1, we slightly abuse the notation by writing w^1_{ud} to represent $w^1_{(n^1_u, n^1_d)}$. Note that elsewhere, the indices of $w^i_{(k,m)}(I)$ are nodes, whereas here refer to letters. To determine the transitions of the packet of flow 1, we check the state of the packet of flow 2 which in this case is unknown. Using the matrix $\mathbf{Q}$ (the scheduler obtains this matrix through a separate mechanism as explained in Section 3) we can calculate the probability of correct decoding for destination δ_1, which is q_{21}. If the decoding fails (it happens with probability $1-q_{21}$), then the scheduler will learn the state of the packet of flow 1, based on the received

feedback from the two destinations. It can be the good state with probability q_{12} (thus moves to the node n_g^1 with probability $q_{12}(1-q_{21})$) or the bad state with probability $1-q_{12}$ (in which case it moves to the node n_b^1 with probability $(1-q_{12})(1-q_{21})$). Thus for the chosen control we get $w_{ud}^1(I) = q_{21}$, $w_{ug}^1(I) = q_{12}(1-q_{21})$, $w_{ub}^1(I) = (1-q_{12})(1-q_{21})$ and all the rest weights are zero. Similarly we develop the link weights for subnetwork $\mathcal{G}_2$ by exchanging 1 and 2.

If the scheduler selects the control $I = \{n_u^1, n_g^2\}$ then the packet of flow 1 will definitely leave the system. Thus now we have $w_{ud}^1(I) = 1$ and the rest weights are zero. Instead if the scheduler selects the control $I = \{n_u^1, n_b^2\}$, we obtain $w_{ud}^1(I) = 0$, $w_{ug}^1(I) = q_{12}$ and $w_{ub}^1(I) = 1 - q_{12}$. Also, given the controls $\{n_g^1, n_u^2\}$ or $\{n_g^1, n_u^2\}$, note that the packet of flow 1 will be either sent to destination or stay at the same node since the packet state is known to the scheduler and cannot change. The weights of the subnetwork $\mathcal{G}_1$ for all controls that activate pairs of queues are given in the Table 1.

6.1.2 Virtual network for F flows and pairwise XOR

The extension of the virtual network mechanism to the case of F flows using pairwise XOR is natural due to the complete feedback information provided. Each packet state is now characterized by the scheduler knowledge about the overhearing vector that contains information about packet knowledge in all $F-1$ destinations. Thus, instead of having three states as before, now we have 3^{F-1} states. Correspondingly, each of the F independently created subnetworks will have $3^{F-1}+1$ nodes, one for each state and one for the destination. We use the generalized notation for the nodes $n_{\mathbf{v}}^i$, where $i \in \mathcal{F}$ indicates the flow and $\mathbf{v}$ is a ternary overhearing vector taking values in $\{u, g, b\}^F$ with $v(i) = g$ by convention (in the previous subsection and Figures 5, 6 we have chosen to omit this element for presentation simplicity). See Figure 6 for an example with $F = 3$. The routing of packets in this network follows exactly the same rules as in the 2–flows case. Specifically, when combining two packets, say belonging to flow i and $j \neq i$, the routing is determined by the elements $v_j(i)$ and $v_i(j)$ correspondingly and by the overhearing probabilities q_{ji} and q_{ij}. Also, in case of failure, only the states $v_i(j)$ and $v_j(i)$ become affected. An exception to this is when the physical receiving node is the destination of more than one flow. Since, the overhearing event concerns primarily the physical nodes and not the flows, the feedback information should update all the elements of vector $\mathbf{v}$ that correspond to flows that have as destination this given node. Note that this complicates the construction of the virtual network but does not affect the analysis for the stability of it. In Figure 6, all possible links are identified for the case of disjoint destinations. Note for example, that $n_{u,u}^1$ is not directly connected with $n_{g,g}^1$ since in order to move from $n_{u,u}^1$ to $n_{g,g}^1$ at least two transmissions are needed.

The problem of scheduling the packets in the original downlink system is mapped to the problem of scheduling the packets in the corresponding virtual network. By representing one queue in the original problem with a subnetwork, we have added another dimension which captures the knowledge obtained progressively about the overhearing states of each packet using the feedback received at the relay. For two packets backlogged in the same virtual node, the scheduler has the same belief and thus these two packets are stochastically equivalent in terms of

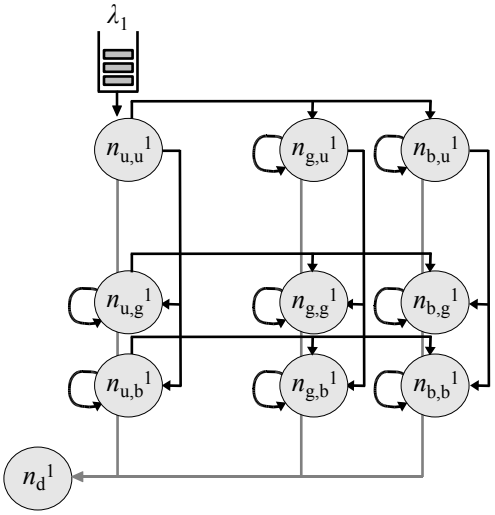

Fig. 6 The virtual subnetwork of flow 1 for the case of 3 flows. Each node represents a possible state of packets of flow 1 as regards the overhearing event in destinations of flows 2, 3, i.e. node $n^1_{b,g}$ holds packets that have been overheard by the destination of flow 3 but not from the one of flow 2. The gray arrows connect all nodes to the destination. The packets enter the state $n^1_{u,u}$ with randomly drawn and unknown overhearing state and navigate until reaching the destination node.

transmission efficiency (i.e. the have the same expected reward), a desired property that is enabled by the use of the virtual network.

6.2 Optimal control of the virtual network for F flows and pairwise XOR

The virtual network developed in Section 6.1.2 differs from the general network treated in [11] in the following important aspect. When a control $I \in \mathcal{I}$ is selected and a packet is chosen for transmission by node j[2], the recipient of the packet is randomly chosen among the outgoing neighbors of node j. Node j does not know the recipient *a priori*, but it knows the probability of the packet ending to a particular outgoing neighbor -this probability depends on the chosen control. Hence the results in [11] cannot be applied directly. However, the methods used in [11] can be extended to analyze the network of interest and to develop an algorithm with maximal stability region. We outline below the main steps of the development.

The virtual network $G = (\mathcal{N}, \mathcal{E})$ developed in 6.1.2 consists of the union of F subnetworks, $G_i = (\mathcal{N}_i, \mathcal{E}_i)$, $i \in \mathcal{F}$. Let $\mathcal{E}^j_o, \mathcal{N}^j_o$ be respectively the set of outgoing links and neighbors of node $j \in \mathcal{N}$, and X_j its backlog. Also, for a given control I, let $w_{(j,k)}(I)$ be the probability that a packet transmitted by node j ends up at node $k \in \mathcal{N}^j_o$.

[2] In the following we will use a simple notation for the nodes. Remember, however, that each node carries information about a flow and a ternary overhearing vector, i.e. the formal name could be for example $n^i_{\mathbf{v}}$.

We assume a slotted system and generalize the network presented in Section 6.1.2 by introducing transmission rates other than 0 and 1 as follows: when control $I \in \mathcal{I}$ is chosen, the maximum number of packets that may be "transmitted" by node j over the link set $\mathcal{E}_o^j$ is $\hat{\mu}_j(I)$. For the model of Section 6.1.2 $\hat{\mu}_j(I)$ is determined as follows.

- If a control I involves transmission from a single node j located at subnetwork i, then
$$\hat{\mu}_k(I) = \begin{cases} r_i^{down} & \text{if } k = j \\ 0 & \text{otherwise} , \end{cases}$$
where r_i^{down} is the maximum number of packets that may be transmitted from the relay to δ_i in a slot.
- If the control involves an XOR packet from nodes j_1, j_2 located at subnetworks $i_1, i_2 \neq i_1$ respectively, then
$$\hat{\mu}_k(I) = \begin{cases} \min\{r_{i_1}^{down}, r_{i_2}^{down}\} & \text{if } k = j_1 \text{or } k = j_2 \\ 0 & \text{otherwise} . \end{cases}$$
- For any other control $\hat{\mu}_k(I) = 0, \forall k \in \mathcal{G}$.
- If a control is taken that requires transmission from a queue which has fewer packets than the specified maximum rates, then *dummy* packets, i.e., packets with id set to zero, are sent instead[3]. This is true also for the case of controls that correspond to transmitting XORs. As we will see in Appendix III this does not give rise to inefficiencies.

We can now present the throughput region of the system. Define the following set of flow variables for each of the network links

$$\mathbf{f} = \{f_e, e \in \mathcal{E}\}.$$

For control $I \in \mathcal{I}$ define the set of vectors $\mathbf{f}$,

$$\Gamma(I) = \left\{\mathbf{f} = \{f_e\} : e = (j,k), f_{(j,k)} = w_{(j,k)}(I)\mu_j : 0 \leq \mu_j \leq \hat{\mu}_j(I), j \in \mathcal{N}, k \in \mathcal{N}_o^j\right\},$$

where μ_j indicates exactly the uncontrollability of where the packet ends up when transmitted from node j. If, for example, one of the possible recipients is a dead end, then we must have $\mu_j = 0$ in order to define the link capacities correctly. Next, define the convex hull of the sets $\Gamma(I)$

$$\mathcal{C} = \text{conv}\left(\Gamma(I), I \in \mathcal{I}\right) \tag{1}$$

Theorem 1 (Throughput Region) *The throughput region of the system is the set of arrival rates $\boldsymbol{\lambda} = \left\{\lambda_j\right\}_{j \in \mathcal{N}}$, $\lambda_j \geq 0$, for which there exists a vector $\mathbf{f} \in \mathcal{C}$ such that for any node $j \in \mathcal{N}$ except the destination nodes it holds*

$$\sum_{e=(k,j) \in \mathcal{E}_o^k} f_e + \lambda_j \leq \sum_{e \in \mathcal{E}_o^j} f_e.$$

[3] The notion of the dummy packet is introduced for technical reasons so the decoding probability for a control does not depend on queue length. Note that XOR controls with dummy packets are dominated by controls without XORs and as such these controls are never applied by our proposed algorithm.

Proof The derivation of the throughput region is based on an extension of the methodology developed in [18]. It is presented in the Appendix I.

Note, that for the problem at hand, the packets always arrive at the nodes with all elements in unknown states, thus the throughput region can be thought as an F-dimensional object.

Algorithm 2: At each decision slot:

1. For each control $I = \{j\}, j \in \mathcal{G}$ form the reward $C(I) = X_j \hat{\mu}_j(I)$.
2. For each control $I = \{i, j\}, i, j \in \mathcal{G}, j \neq i$
   – form the weights

$$c_i(I) = \max\left\{X_i - \sum_{k \in \mathcal{N}_o^i} w_{(i,k)}(I) X_k, 0\right\},$$

$$c_j(I) = \max\left\{X_j - \sum_{k \in \mathcal{N}_o^j} w_{(j,k)}(I) X_k, 0\right\},$$

   – and then the reward

$$C(I) = c_i(I)\hat{\mu}_i(I) + c_j(I)\hat{\mu}_j(I).$$

3. Then select $I^* = \arg\max_{I \in \mathcal{I}}\{C(I)\}$.

Theorem 2 (Optimality of Algorithm 2) *Algorithm 2 is throughput optimal for the system described in Section 6.1.2 and the assumptions 1-5.*

Proof Based on the throughput region described above and using Lyapunov function techniques as in [11], we show in the Appendix II that the algorithm described above has maximal stability region.

A question that can be raised regarding Algorithm 2 efficiency relates to situations where the links permit the transmission of a great number of packets while the queue sizes are small. We give an answer to this question in Appendix III.

7 Numerical results

We have used a discrete event simulator written in Matlab to analyze scheduling performance of the proposed algorithms and compare it to other simple heuristic algorithms used in practical systems. We study the case of $F = 2$ flows with all links having unit rate (1 packet/slot) and probabilities of overhearing q_{12} and q_{21}. We generate packets using a Poisson process with rate λ_1 and λ_2 correspondingly for the two flows and we run the simulation for $50k$ time slots.

In addition to algorithms 1 and 2, we are interested in algorithms that are not based on queue information.

1. Deterministic (det): The scheduler identifies the HOL packets and using deterministic overhearing information selects the largest possible combination of packets that can be decoded. The ties are resolved randomly.

2. Deterministic advanced (det_adv): Similar with deterministic, with the difference that the scheduler can select among any packet in the queue. This algorithm avoids the well known problem of input-queued switches called HOL blocking, [17]. This policy is expected to have the best performance since it utilizes deterministic knowledge of overhearing and knowledge of arrival rates.
3. Statistical (sta): If both q_{12} and q_{21} are larger than 0.8, the XOR combination of the HOL packets is transmitted, otherwise one of the packets in the queues is chosen at random for transmission. Note that this threshold policy has been proposed in previous works as a simple suboptimal heuristic.

We compare the algorithms by plotting the average aggregate queue backlogs of the system when each algorithm is in use after $50k$ slots. As long as the backlogs are retained small, the system is stable. We plot the average aggregate backlogs for several values of λ_1 and we set $\lambda_2 = \alpha\lambda_1$ where α is a scaling constant defined differently in each setting.

In Figures 7-a and 7-b we set $q_{12} = q_{21} = 0.6$ and $\alpha = 1$ on Figure 7-a, $\alpha = 0.5$ on Figure 7-b. We call this case symmetric, since both flows have same chances of being overheard by the pair destination. First, a direct comparison can be made between Algorithm 1 and det_adv. Both algorithms decide using deterministic information of overhearing events and indeed they appear better than the rest of the algorithms. The difference between the two algorithms is that Algorithm 1 is based on queue information and it is provably throughput optimal. The algorithm det_adv will always encode if an opportunity appears, but the queue agnostic approach it is using is expected to bring a small deterioration in the performance in dynamic (in terms of arrival rate for example) scenarios. Nevertheless, for the static scenarios shown, the two algorithms are quite similar in performance and the simulation granularity does not allow for distinguishing between the two. Both can be thought as yardsticks for the rest of the algorithms.

In order to compare the performance of two algorithms, one can decide on a limit for the backlogs (say 50) and compare the arrival rate for which the two algorithms exceed this value. We call the difference between these two values the throughput performance gap. The gap between algorithm det and the det_adv algorithm provides an estimation of the efficiency loss due to HOL blocking. This loss seems to be higher when the arrival rates are not equal (it is 0.042 in Figure 7-a and 0.12 in Figure 7-b).

The gap between Algorithm 2 and Algorithm 1 depicts the efficiency loss due to the use of statistical information instead of deterministic one (it is 0.045 in Figure 7-a and 0.066 in Figure 7-b). This can be also compared to the gap between these algorithms and the sta algorithm which in this scenario behaves as if NC is disabled. For example, we may observe that the performance gap from not using NC is +0.12 (24%) for relying only on statistical information (Algorithm 2) and +0.165 (33% corresponding to the improvement of λ_1) when having deterministic overhearing information (Algorithm 1) in Figure 7-a. Despite the fact that the lack of deterministic information brings reduction in throughput, it is evident that a throughput gain due to NC is still maintained even for relatively small overhearing probabilities and this is an encouraging result. Using the statistical overhearing

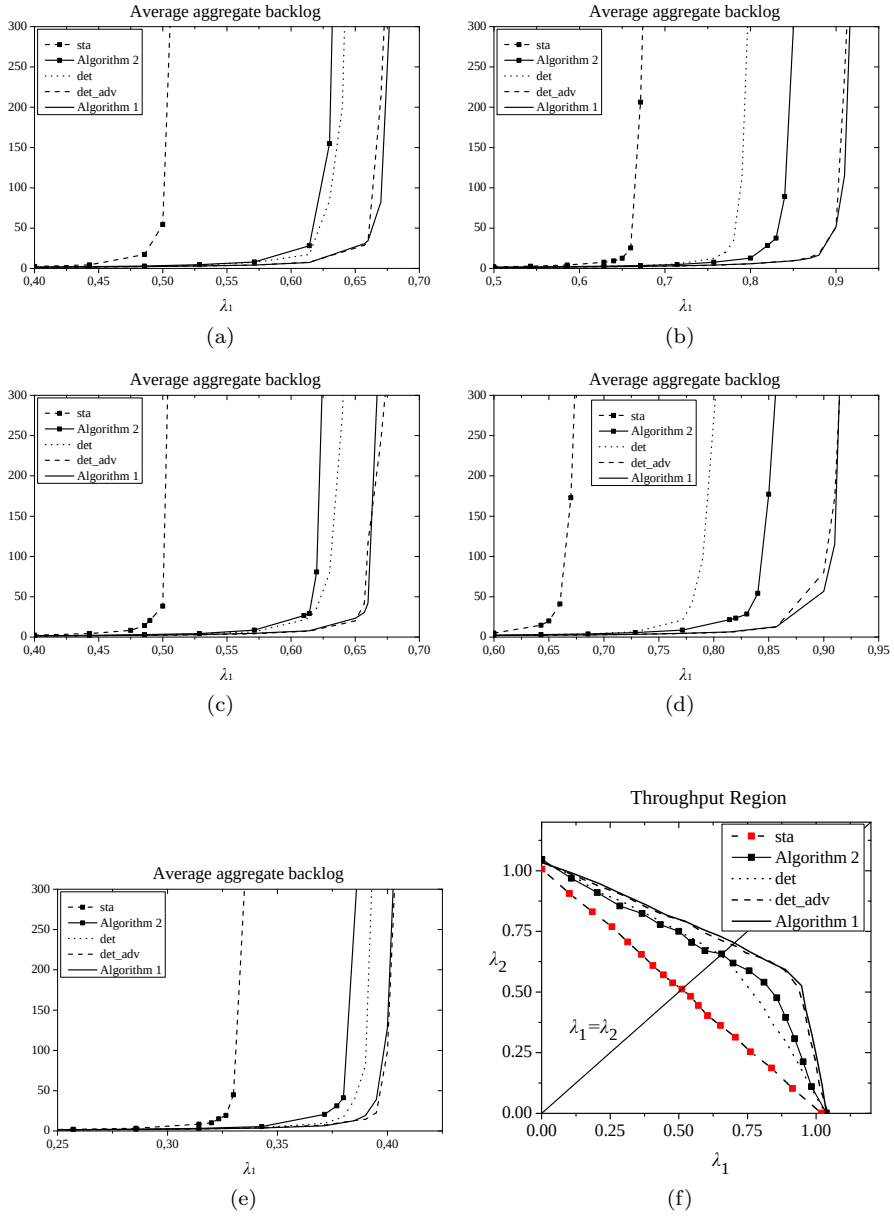

Fig. 7 (a)–(e) Average aggregate backlog. (a) Symmetric case $q_{12} = 0.6$ and $q_{21} = 0.6$; $\lambda_2 = \lambda_1$. (b) Symmetric case $q_{12} = 0.6$ and $q_{21} = 0.6$; $\lambda_2 = 0.5\lambda_1$. (c) Asymmetric case $q_{12} = 0.5$ and $q_{21} = 0.8$; $\lambda_2 = \lambda_1$. (d) Asymmetric case $q_{12} = 0.5$ and $q_{21} = 0.8$; $\lambda_2 = 0.5\lambda_1$. (e) Asymmetric case $q_{12} = 0.5$ and $q_{21} = 0.8$; $\lambda_2 = 2\lambda_1$. (f) Throughput region measured such that average backlog is always smaller than 500. Asymmetric case $q_{12} = 0.5$ and $q_{21} = 0.8$;

information is well motivated by these Figures. Also, using a fixed threshold (e.g. 0.8) is clearly not a good solution.

In Figures 7-c,d,e we set $q_{12} = 0.5$ and $q_{21} = 0.8$ and thus make the decoding of flow 1 more probable than that of flow 2 (asymmetric case). We further impose $\alpha = 1$ in Figure 7-c, $\alpha = 0.5$ in Figure 7-d and $\alpha = 2$ in Figure 7-e for the relation of arrival rates between the two flows. The gain of Algorithm 2 versus the no NC case (algorithm sta) is +0.118 (23%), +0.179 (27%) and +0.005 (15.1%) in the three Figures respectively. Similarly the gain of Algorithm 1 is +0.163 (32.5%), +0.236 (35.7%) and +0.07 (21%). The performance gap is always retained above 20%, which demonstrates the ability of queue-based algorithms to adapt to asymmetric scenarios and unknown arrival rates. Note that the accuracy of the results when the backlogs are very high is not satisfactory, thus the rippling between the Algorithm 1 and det_adv lines in Figure 7-c. We interpret this rippling as a statistical error and we stress that for accurate comparison the range of values (0-100) should be used, where we provide more detail (more figure points).

The above comparison is made clear in the last Figure, 7-f, where we demonstrate the stability region of all algorithms for the case of $q_{12} = 0.5$ and $q_{21} = 0.8$. A point in this region is found by fixing α and making a binary search with accuracy 0.001. A point (λ_1, λ_2) is considered stable if after 25000 slots, all flow backlogs (the sum of backlogs of the queues belonging to one flow) remain smaller than 500. The area separating Algorithm 1 and Algorithm 2 is the regret region and describes the loss in throughput due to lack of deterministic knowledge about the state.

8 Conclusions

In this paper we addressed the problem of downlink scheduling when pairwise XORing of packets is allowed and there is statistical overhearing information and feedback at the relay. We developed a virtual network approach to address the problem at hand. This virtual network was used to provide a stabilizing backpressure policy. In this work the statistical overhearing information is assumed to be provided to the relay by an independent mechanism. It seems possible to use the received feedback to estimate the overhearing probabilities along the lines of [19], however the exact manner this can be done and the potential benefits need to be further investigated. Future work of interest pertains to generalizing the model, considering uplink and downlink jointly as well as multihop scheduling, allowing for storing of encoded packets, including channel erasures and developing approximation algorithms for the case of $|\mathcal{P}| > 2$, see Appendix IV.

The model of the virtual network is a versatile model which can be extended in many ways. For the wireless NC problem, an interesting extension left for future work is to study the problem when some of the Assumptions 1-5 in Section 3 are relaxed. Such a task can be performed in a two step process. First, one needs to identify the appropriate definition of a virtual network that captures accurately all the aspects of the actual system. Then, the approach of Section 6 can be applied to provide the throughput region and a throughput optimal algorithm. While this programme seems promising, several intricacies arise when one takes a closer look at relaxing the assumptions; special constraints on admissible controls arise and in certain cases the virtual network contains an infinite number of nodes, the study of which may require the development of new techniques.

Finally, our model assumes that $s^i(i) = 0$, i.e., the destination of a flow cannot overhear the source. We introduced this assumption in order to simplify the discussion. If the assumption is relaxed, then two cases can be considered:

1. The relay knows deterministically whether each receiver has overheard a packet destined to itself. Then it can place in the queues only the packets that are not overheard and the model is not affected.
2. The relay knows statistically whether each receiver has overheard a packet destined to itself. This case can also be dealt with the method presented in this paper, by enhancing the feedback mechanism. Specifically an extra bit can be added in each ACK message when an XOR packet is sent, indicating the above knowledge, i.e. whether the ACK was due to successful decoding having the key or because of a direct overhearing from the source.

The study of the case where there is the possibility of retransmission from the source or from the relay involves a more complicated scheduling problem and is left as future work.

Acknowledgements We would like to thank the anonymous reviewers for their valuable contribution to the improvement of this work.

References

1. MIT Roofnet. http://pdos.csail.mit.edu/roofnet.
2. NCRAWL experiments. http://nitlab.inf.uth.gr/NITlab/index.php/ncrawl-experiments/results.html.
3. R. Ahlswede, N. Cai, S-Y. R.Li, and R. W. Yeung. Network information flow. In *IEEE Trans. Inform. Theory, pp. 1204-1216*, July 2000.
4. I. Broustis, G. S. Paschos, D. Syrivelis, L. Georgiadis, and L. Tassiulas. NCRAWL: Network Coding for Rate Adaptive Wireless Links. arXiv:1104.0645.
5. P. Chaporkar and A. Proutiere. Adaptive Network Coding and Scheduling for Maximizing Throughput in Wireless Networks. In *ACM MOBICOM*, 2007.
6. P. Chaporkar, A. Proutiere, H. Asnani, and A. Karandikar. Scheduling with limited information in wireless systems. In *ACM MobiHoc*, pages 75–84, 2009.
7. M.A.R. Chaudhry and A. Sprintson. Efficient algorithms for index coding. In *IEEE INFOCOM*, 2008.
8. A. Eryilmaz and D. S. Lun. Control for inter-session network coding. In *Proc. of ITA Workshop*, February 2007.
9. C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul. Wireless network coding: Opportunities & Challenges. In *IEEE Military Communications Conference 2007*, October 2007.
10. A. Fu, E. Modiano, and J.N. Tsitsiklis. Optimal transmission scheduling over a fading channel with energy and deadline constraints. *IEEE Trans. on Wireless Communications*, 5(3):630–641, 2006.
11. L. Georgiadis, M. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 1:1–147, 2006.
12. A J. Goseling, R. Matsumoto, T. Uyematsu, and J. H. Weber. Lower Bounds on the Maximum Energy Benefit of Network Coding for Wireless Multiple Unicast. *EURASIP Journal on Wireless Communications and Networking (Special Issue on Wireless Network Coding)*, (Article ID 605421), 2010.
13. T. Ho, Y. Chang, and K. J. Han. On constructive network coding for multiple unicasts. In *Proc. of 44th Allerton conference on Communication, Control and Computing*, September 2006.
14. T. Ho and H. Viswanathan. Dynamic algorithms for multicast with intra-session network coding. *IEEE Trans. Inf. Theor.*, 55:797–815, February 2009.

15. K. Jagannathan, S. Mannor, I. Menache, and E. Modiano. A state action frequency approach to throughput maximization over uncertain wireless channels. In *the 30th IEEE International Conference on Computer Communications, INFOCOM 2011*, pages 491–495, 2011.
16. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in The Air: Practical Wireless Network Coding. In *ACM SIGCOMM*, 2006.
17. A. Mekkittikul and N. McKeown. A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches. In *IEEE Infocom*, pages 792–799, April 1998.
18. E. Modiano M.J. Neely and C. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications*, 23:89–103, January 2005.
19. M. Neely. Max weight learning algorithms with application to scheduling in unknown environments. In *Information Theory and Applications Workshop*, pages 240–249, 2009.
20. M. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. Ph.D. Dissertation, Massachusetts Institute of Technology, LIDS. November 2003.
21. M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Trans. Inf. Theor.*, 52:2915–2934, July 2006.
22. G. S. Paschos, L. Georgiadis, and L. Tassiulas. Optimal scheduling of pairwise XORs under statistical overhearing and feedback. In *RAWNET workshop: Workshop on Resource Allocation and Cooperation in Wireless Networks, WiOPT*, April 2011.
23. S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta. Loss-Aware Network Coding for Unicast Wireless Sessions: Design, Implementation, and Performance Evaluation. In *ACM SIGMETRICS*, 2008.
24. E. Rozner, A.P. Iyer, Y. Mehta, L. Qiu, and M. Jafry. ER: Efficient Retransmission Scheme for Wireless LANs. In *ACM CONEXT*, 2007.
25. B. Scheuermann, W. Hu, and J. Crowcroft. Near-Optimal Co-ordinated Coding in Wireless Multihop Networks . In *ACM CONEXT*, 2007.
26. A. L. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50:401–457, July 2005.
27. L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37:1936–1948, 1992.
28. L. Ying and S. Shakkottai. On throughput optimality with delayed network-state information. In *Information Theory and Applications Workshop*, pages 339–344, 2008.

Appendix I

In this Appendix we give the proof for the necessity of the conditions of Theorem 1. The sufficiency of the the conditions follows directly from the proof of Theorem 2 which shows that Algorithm 2 stabilizes the system for any arrival rate vector that is in the interior of the throughput region.

The proof of this Section is based on the technique used in [10]. We will need the following lemmas.

Lemma 1 *If a random vector* $\mathbf{X}(t)$ *converges to* $\mathbf{X}$ *on a set* A *then for any* $\epsilon > 0$

$$\lim_{t\to\infty} \mathrm{P}(\{|\mathbf{X}(t) - \mathbf{X}| \leq \epsilon\} \cap A) = \mathrm{P}(A).$$

Lemma 2 *Consider two sequences of real numbers* a_t, b_t $t = 1, 2, \ldots$. *Then*

$$\limsup_{t\to\infty} (a_t + b_t) \geq \limsup_{t\to\infty} a_t + \liminf_{t\to\infty} b_t.$$

Lemma 3 *If a network is stable then*

$$\lim_{V\to\infty} \liminf_{t\to\infty} \mathrm{P}\left(\sum_{j\in\mathcal{N}} X_j(t) > V\right) = 0.$$

Lemmas 1 and 2 can be found in standard textbooks while a proof of Lemma 3 can be found in [20].

Stability Region: we repeat here the definition of the stability region as in Theorem 1.

– Let $\boldsymbol{f}$ denote a vector
$$\boldsymbol{f} = \{f_e : e \in \mathcal{E}\}$$
and for $I \in \mathcal{I}$, let $\Gamma(I)$ be the set of vectors $\boldsymbol{f}$ defined as
$$\Gamma(I) = \left\{\boldsymbol{f} :\ e = (j,k) \in \mathcal{E}_o^j,\ f_e = w_e^j(I)\mu_j,\ 0 \le \mu_i \le \hat{\mu}_i(I),\ i \in \mathcal{N}\right\}. \quad (2)$$
Note that that $\Gamma(I)$ is convex.
– Let
$$\mathcal{C} = \left\{\boldsymbol{f} :\ \boldsymbol{f} = \sum_{I\in\mathcal{I}} \phi_I \boldsymbol{f}_I,\ \boldsymbol{f}_I \in \Gamma(I),\ \sum_{I\in\mathcal{I}} \phi_I = 1\right\}.$$
Since $\Gamma(I),\ I \in \mathcal{I}$ are convex it can be shown that
$$\mathcal{C} = \text{conv}\{\Gamma(I) : I \in \mathcal{I}\},$$
where conv($\mathcal{A}$) denotes the convex hull of $\mathcal{A}$.
– The stability region is the set of arrival rates $\boldsymbol{\lambda} = \{\lambda_i\}_{i\in\mathcal{N}}$ for which there exists a vector $\boldsymbol{f}$ in $\mathcal{C}$ such that for any node $i \in \mathcal{N}$
$$\sum_{e\in\mathcal{E}_{in}^i} f_e + \lambda_i \le \sum_{e\in\mathcal{E}_o^i} f_e.$$

For the proof we consider a somewhat more general network than the one presented in Section 6. Specifically, we consider a network consisting of $\mathcal{N} \cup d$ nodes and $\mathcal{E}$ links, where the special node d represents the destination of traffic originated at the other nodes in $\mathcal{N}$. Let $\mathcal{E}_o^i$ represent the set of outgoing links from node i and $\mathcal{E}_{in}^i$ set of incoming links. A *finite* set of controls $\mathcal{I}$ is available. For each control $I \in \mathcal{I}$, "transmission" takes place over the sets of outgoing links of node $i \in \mathcal{N}$, $\mathcal{E}_o^i$, as follows.

– If at a given slot control $I \in \mathcal{I}$ is applied, then for any node $i \in \mathcal{N}$ at most $\hat{\mu}_i(I) \ge 0$ packets may be transmitted over the set $\mathcal{E}_o^i$ in the following random manner.
– For each $i \in \mathcal{N}$ and $I \in \mathcal{I}$ there is a random sequence $\left\{R_n^i(I)\right\}_{n=1}^{\infty}$, where each $R_n^i(I)$ takes values in the set $\mathcal{E}_o^i$. The nth packet transmitted over the set $\mathcal{E}^i$ when control I is applied, is received *only* by the recipient of the link $R_n^i(I)$.

For a given n and I, the random variables $\left\{R_n^i(I)\right\}_{i\in\mathcal{N}}$ may be arbitrarily correlated. Moreover the random sequences $\left\{R_n^i(I)\right\}_{n=1}^{\infty}$ are obeying the strong law of large numbers, i.e., for any $I \in \mathcal{I}$, $i \in \mathcal{N}$,

$$\lim_{n\to\infty} \frac{\sum_{m=1}^{n} \mathbb{1}\left[R_m^i(I) = e\right]}{n} = w_e^i(I) \text{ a.e., } e \in \mathcal{E}_o^i(I), \tag{3}$$

$$\sum_{e\in\mathcal{E}^i} w_e^i(I) = 1. \tag{4}$$

Packets may arrive at each node $i \in \mathcal{N}$ and must be delivered to destination node d. We denote by $A_i(t)$ the number of arrivals up to time t. We assume that

$$\lim_{t\to\infty} \frac{A_i(t)}{t} = \lambda_i,\ i \in \mathcal{N}.$$

If in the virtual network developed in Section 6 all destination nodes n_d^i are replaced by a single node d, maintaining the links of the destination nodes with the rest of the nodes in $\mathcal{N}$, the resulting network is a special case of the network model defined here. We mention that the model presented here can be easily generalized to include channel states and multi commodity flows but we opted for the current description since it suffices for our purposes and avoids further complicated notation.

In the following, to avoid trivial cases we assume that there is a path from every node $i \in \mathcal{N}$ to d (this is needed in the sufficient conditions to justify the existence of an interior point).

Assume that a policy π stabilizes the network for a rate vector $\{\lambda_i\}_{i\in\mathcal{N}}$. Let the system operate under this policy and define the following random variables

- $\mathcal{W}(I,t)$: subset of time slots in $\{1,...,t\}$ when control I is applied. Let $W(I,t) = |\mathcal{W}(I,t)|$.
- $F^i(I,t)$: the number of packets transmitted up to time t over set $\mathcal{E}_o^i$, when control I is applied.
- $F_e(t)$: total number of packets transmitted over link e up to time t.

According to the definitions above, it holds

$$\sum_{I\in\mathcal{I}} W(I,t) = t \tag{5}$$

and for any $e = (i,k) \in \mathcal{E}^i$,

$$F_e(t) = \sum_{I\in\mathcal{I}} \sum_{m=1}^{F^i(I,t)} \mathbb{1}\left[R_m^i(I) = e\right], \tag{6}$$

where we define $\sum_{m=1}^{0} X_i = 0$.

Let $X_i(t)$ be the number of packets at node i at time t. Assuming that the queues are initially empty, the following holds for any node i other than the destination node.

$$A_i(t) + \sum_{e\in\mathcal{E}_{in}^i} F_e(t) - \sum_{e\in\mathcal{E}_o^i} F_e(t) = X_i(t). \tag{7}$$

The rest of the proof involves several technical details. To clarify the steps, we provide first an outline of the proof making some simplifying assumptions. Specifically, assume that under policy π, for any $I \in \mathcal{I}$,

1. The (long term) proportion of time when control I is applied is well defined and positive, (hence $\lim_{t\to\infty} W(I,t) = \infty$),

$$\lim_{t\to\infty} \frac{W(I,t)}{t} = \phi_I > 0.$$

2. The average number of packets transmitted when control I is applied is well defined,

$$\lim_{t\to\infty} \frac{F^i(I,t)}{W(I,t)} = \mu_i(I).$$

We note that these assumptions do not hold for all policies and thus some ill-behaving policies are left outside this consideration. The technical details that complicate the proof aim at removing these assumptions.

We have from (5),

$$\sum_{I\in\mathcal{I}} \phi_I = 1$$

and from (3), (4), (6), for $e = (i,k)$

$$\begin{aligned} f_e &\triangleq \lim_{t\to\infty} \frac{F_e(t)}{t} \\ &= \lim_{t\to\infty} \sum_{I\in\mathcal{I}} \frac{W(I,t)}{t} \frac{F^i(I,t)}{W(I,t)} \sum_{m=1}^{F^i(I,t)} \frac{1\left[R^i_m(I) = e\right]}{F^i(I,t)} \\ &= \sum_{I\in\mathcal{I}} \phi_I \left(\mu_i(I) w^i_e(I)\right) \end{aligned} \tag{8}$$

Since at most $\hat{\mu}_i(I)$ packets can be transmitted in each slot when control I is applied, it holds

$$0 \le \mu_i(I) \le \hat{\mu}_i(I).$$

From this and (8) we see that the vector $\{f_e\}_{e\in\mathcal{E}}$ belongs to the region $\mathcal{C}$. Moreover, dividing both sides of (7) by t, taking limit as $t \to \infty$and using the fact that for a stabilizing policy $\lim_{t\to\infty} X_i(t)/t = 0$, we have from (7),

$$\lambda_i + \sum_{e\in\mathcal{E}^i_{in}} f_e = \sum_{e\in\mathcal{E}^i_0} f_e, \text{ for } i \in \mathcal{N}$$

which implies that the arrival rate vector belongs to the Stability region as claimed.

We now proceed with the detailed proof.

In the following Ω denotes the underlying probability space. Let $\mathcal{P}_\mathcal{T} = \left(\mathcal{T}_a, \mathcal{T}_f\right)$ denote a partition of a set $\mathcal{T}$, i.e., $\mathcal{T} = \mathcal{T}_a \cup \mathcal{T}_f$, $\mathcal{T}_a \cap \mathcal{T}_f = \emptyset$, and denote by $\mathbb{P}_\mathcal{T}$ the set of these partitions. We need the following preliminary lemmas.

Lemma 4 *Let the stabilizing policy π be applied. Then there is a partition $\mathcal{P}^o_\mathcal{I} = \left(\mathcal{I}^o_a, \mathcal{I}^o_f\right)$ of the control set $\mathcal{I}$, with $\mathcal{I}^o_a \neq \emptyset$, such that the set*

$$\Omega_0 = \left\{\omega : \lim_{t\to\infty} W(I,t) = \infty,\ I \in \mathcal{I}^o_a,\ \lim_{t\to\infty} W(I,t) < \infty,\ I \in \mathcal{I}^o_f\right\}$$

has positive probability.

Moreover, there are partitions of the set $\mathcal{I}_a^o$, $\left\{\mathcal{P}^i_{\mathcal{I}_a^o}\right\}_{i\in\mathcal{N}} = \left\{\left(\mathcal{I}_a^{o,i}, \mathcal{I}_f^{o,i}\right)\right\}_{i\in\mathcal{N}}$ *such that the subset of* Ω_0,

$$\Omega_1 = \left\{\omega\in\Omega_o : \lim_{t\to\infty} F^i(I,t) = \infty,\ I\in\mathcal{I}_a^{o,i},\ \lim_{t\to\infty} F^i(I,t) < \infty,\ I\in\mathcal{I}_f^{o,i},\ i\in\mathcal{N}\right\}$$

has positive probability.

Proof Since the sequence $W(I,t)$ is nondecreasing in t it converges either to a finite number or to infinity. Hence defining

$$\Omega_T = \left\{\omega : \lim_{t\to\infty} W(I,t) \text{ exists}\right\},$$

we have that $\mathrm{P}(\Omega_T) = 1$. Define now

$$\Omega_{\mathcal{P}_\mathcal{I}} = \left\{\omega : \lim_{t\to\infty} W(I,t) = \infty,\ I\in\mathcal{I}_a, \lim_{t\to\infty} W(I,t) < \infty,\ I\in\mathcal{I}_f\right\}.$$

Then,

$$\Omega_T = \cup_{\mathcal{P}_\mathcal{I}\in\mathbb{P}_\mathcal{I}}\Omega_{\mathcal{P}_\mathcal{I}}.$$

Since $\mathrm{P}(\Omega_T) > 0$, one of the sets on the right of the last equality must have nonzero probability and this establishes the existence of Ω_0. Moreover, if $\mathcal{I}_a = \emptyset$ then (7) on Ω_0 we would have

$$\lim_{t\to\infty}\sum_{I\in\mathcal{I}} W(I,t) < \infty,$$

which contradicts (5).

Decomposing Ω_0 in a similar fashion based on the existence of limits of the sequences $F^i(I,t)$, establishes the second part of the lemma.

Lemma 5 *Let the stabilizing policy* π *be applied. Then there is a partition* $\mathcal{P}_\mathcal{I}^o = \left(\mathcal{I}_a^o, \mathcal{I}_f^o\right)$ *of the control set* $\mathcal{I}$, *with* $\mathcal{I}_a^o \neq \emptyset$ *and partitions of the set* $\mathcal{I}_a^o$, $\left\{\mathcal{P}^i_{\mathcal{I}_a^o}\right\}_{i\in\mathcal{N}} = \left\{\left(\mathcal{I}_a^{o,i}, \mathcal{I}_f^{o,i}\right)\right\}_{i\in\mathcal{N}}$, *such that: for any* $\epsilon > 0$ *there is*

a) a realization $\omega\in\Omega$,

b) a t *large enough,*

for which that the following hold

$$\frac{X_i(t)}{t} \leq \epsilon,$$

$$\left|\frac{\sum_{m=1}^{F^i(I,t)} \mathbb{1}\left[R_m^i(I) = e\right]}{F^i(I,t)} - w_e^i(I)\right| \leq \epsilon,\ I\in\mathcal{I}_a^{o,i},\ i\in\mathcal{N}, \tag{9}$$

$$0 \leq \frac{F^i(I,t)}{W(I,t)} \leq \epsilon,\ I\in\mathcal{I}_f^{o,i},\ i\in\mathcal{N}, \tag{10}$$

$$0 \leq \frac{W(I,t)}{t} \leq \epsilon,\ I\in\mathcal{I}_f^o, \tag{11}$$

$$\lambda_i - \epsilon \leq \frac{A_i(t)}{t},\ i\in\mathcal{N}. \tag{12}$$

Proof Consider the set Ω_1 of Lemma 4. For any realization belonging to this set, since $\lim_{t\to\infty} F^i(I,t) = \infty$ for $I \in \mathcal{I}_a^{o,i}$, $i \in \mathcal{N}$ we have that

$$\lim_{t\to\infty} \frac{\sum_{m=1}^{F^i(I,t)} \mathbb{1}\left[R_m^i(I) = e\right]}{F^i(I,t)} = w_e^i(I),\ I \in \mathcal{I}_a^{o,i},\ i \in \mathcal{N}. \tag{13}$$

Also, since $\lim_{t\to\infty} F^i(I,t) < \infty$ and $\lim_{t\to\infty} W(I,t) = \infty$ for $I \in \mathcal{I}_f^{o,i}$, it holds

$$\lim_{t\to\infty} \frac{F^i(I,t)}{W(I,t)} = 0,\ I \in \mathcal{I}_f^{o,i},\ i \in \mathcal{N} \tag{14}$$

and similarly,

$$\lim_{t\to\infty} \frac{W(I,t)}{t} = 0,\ I \in \mathcal{I}_f^{o}. \tag{15}$$

By the ergodicity of the arrival process,

$$\lim_{t\to\infty} \frac{A_i(t)}{t} = \lambda_i,\ i \in \mathcal{N} \tag{16}$$

Defining the set Ω_ϵ(t) as the subset where (9) - (12) hold simultaneously, it follows from (13)-(16) and Lemma 1 that

$$\liminf_{t\to\infty} \mathrm{P}(\Omega_\epsilon(t) \cap \Omega_1) = \mathrm{P}(\Omega_1) > 0.$$

Hence for any fixed V,

$$\limsup_{t\to\infty} \mathrm{P}\left(\left\{\sum_{j\in\mathcal{N}} X_j(t) \le V\right\} \cap \Omega_\epsilon(t) \cap \Omega_1\right) \tag{17}$$

$$\ge \limsup_{t\to\infty} \left(\mathrm{P}\left(\left\{\sum_{j\in\mathcal{N}} X_j(t) \le V\right\}\right) + \mathrm{P}(\Omega_\epsilon(t) \cap \Omega_1) - 1\right)$$

$$= \limsup_{t\to\infty} \mathrm{P}\left(\left\{\sum_{j\in\mathcal{N}} X_j(t) \le V\right\}\right) + \mathrm{P}(\Omega_1) - 1, \tag{18}$$

where the equality follows from Lemma 2.

From Lemma 3 it follows that

$$\lim_{V\to\infty} \limsup_{t\to\infty} \mathrm{P}\left(\sum_{j\in\mathcal{N}} X_j(t) \le V\right) = 1.$$

Hence we can pick V large enough so that

$$\limsup_{t\to\infty} \mathrm{P}\left(\sum_{j\in\mathcal{N}} X_j(t) \le V\right) \ge 1 - \mathrm{P}(\Omega_1)/2.$$

For this choice of V it follows from (18) that

$$\limsup_{t\to\infty} \mathrm{P}\left(\left\{\sum_{j\in\mathcal{N}} X_j(t) \le V\right\} \cap \Omega_\epsilon(t) \cap \Omega_1\right) \ge \mathrm{P}(\Omega_1)/2 > 0.$$

Therefore there is a sequence of times t_m, with $\lim_{m\to\infty} t_m = \infty$ such that

$$\mathrm{P}\left(\left\{\sum_{j\in\mathcal{N}} X_j\left(t_m\right) \le V\right\} \cap \Omega_\epsilon(t_m) \cap \Omega_1\right) > 0. \tag{19}$$

Picking a t_m large enough so that $V/t_m \le \epsilon$ we conclude from (19) that there is a realization satisfying all the conditions of the lemma. □

Proof of Theorem 1 (Necessity): For any $\epsilon > 0$ and for the corresponding realization of Lemma 5 define

$$f_e^\epsilon = \frac{F_e(t)}{t},\ e \in \mathcal{E},$$
$$\mu_i^\epsilon(I) = \frac{F^i(I,t)}{W(I,t)},\ i \in \mathcal{N},\ I \in \mathcal{I}_f^o,$$
$$\phi^\epsilon(I) = \frac{W(I,t)}{t}.$$

These quantities are all bounded, specifically,

$$0 \le f_e^\epsilon \le \max_{I\in\mathcal{I}} \hat{\mu}_i(I),\ e = (i,k),$$
$$0 \le \mu_i^\epsilon(I) \le \max_{I\in\mathcal{I}} \hat{\mu}_i(I),$$
$$0 \le \phi^\epsilon(I) \le 1.$$

Moreover, by (5) it holds

$$\sum_{I\in\mathcal{I}} \phi^\epsilon(I) = 1. \tag{20}$$

It then follows from (6) that for $e = (i,k) \in \mathcal{E}$ it holds

$$\begin{aligned} f_e^\epsilon =& \frac{F_e\left(t\right)}{t} \\ =& \sum_{I\in\mathcal{I}_a^{o,i}} \frac{W(I,t)}{t} \frac{F^i(I,t)}{\mathcal{W}^I(t)} \frac{\sum_{m=1}^{F^i(t)} \mathbb{1}\left[R_m^j\left(I\right) = e\right]}{F^j(I,t)} \\ &+ \sum_{I\in\mathcal{I}_f^{o,i}} \frac{W(I,t)}{t} \frac{\sum_{m=1}^{F^i(t)} \mathbb{1}\left[R_m^j\left(I\right) = e\right]}{W(I,t)} \\ &+ \sum_{I\in\mathcal{I}_f^{o}} \frac{\sum_{m=1}^{F^i(t)} \mathbb{1}\left[R_m^j\left(I\right) = e\right]}{t}. \end{aligned}$$

Since $\sum_{m=1}^{F^i(I,t)} \mathbb{1}\left[R_m^j\left(I\right) = e\right] \le \hat{\mu}_i(I) W(I,t)$, using the definitions and the inequalities in Lemma 5 we have

$$\begin{aligned} f_e^\epsilon \le& \sum_{I\in\mathcal{I}_a^{o,i}} \phi^\epsilon(I)\mu_i^\epsilon(I)\left(w_e^i(I) + \epsilon\right) \\ &+ \sum_{I\in\mathcal{I}_f^{o,i}} \phi^\epsilon(I)\epsilon \\ &+ |\mathcal{I}|\, M\epsilon, \end{aligned} \tag{21}$$

where $M = \max_{I\in\mathcal{I}} \max_{i\in\mathcal{N}} \hat{\mu}_i(I)$. Similarly,

$$f_e^\epsilon \geq \sum_{I\in\mathcal{I}_a^{o,i}} \phi^\epsilon(I)\mu_i^\epsilon(I)\left(w_e^i(I)-\epsilon\right) \tag{22}$$

and taking into account (7),

$$\lambda_i - \epsilon + \sum_{e\in\mathcal{E}_{in}^i} f_e^\epsilon - \sum_{e\in\mathcal{E}_o^i} f_e^\epsilon \leq \epsilon. \tag{23}$$

Consider now the sequence of vectors

$$\boldsymbol{V}_n = \left\{ f_e^{1/n},\, e\in\mathcal{E},\, \phi^{1/n}(I),\, I\in\mathcal{I},\, \mu_i^{1/n},\, i\in\mathcal{N}\right\},\, n=1,\ldots$$

This sequence is bounded and hence it contains a convergent subsequence $\boldsymbol{V}_{n_k}, k = 1,...$ Let $\{f_e,\, e\in\mathcal{E},\, \phi(I),\, I\in\mathcal{I},\, \mu_i\}$ be the limit of this subsequence. Taking limits in (20), (21)-(23) we see that using this limit sequence shows that $\{\lambda_i\}_{i\in\mathcal{N}} \in \mathcal{C}$. □

Appendix II

In this Appendix we give the proof of Theorem 2.

Proof of Theorem 2: Define the following quadratic Lyapunov function

$$L(\mathbf{X}) = \sum_{i\in\mathcal{N}} X_i^2$$

and the following conditional expectation, called *Lyapunov drift*

$$\Delta\mathbf{X}(\tau) \doteq \mathrm{E}\{L(\mathbf{X}(\tau+1)) - L(\mathbf{X}(\tau))|\mathbf{X}(\tau)\}.$$

We would like to show the system is stable under Algorithm 2 whenever the arrival vector $\boldsymbol{\lambda}$ lies inside the region of Theorem 1. For this, it is enough to show that the Lyapunov drift is negative whenever the backlogs are large enough, i.e. that there exist positive constants B, ξ such that for all τ

$$\Delta\mathbf{X}(\tau) \leq B - \xi \sum_{i\in\mathcal{N}} X_i(\tau),$$

see [11] and in particular Lemma 4.1. In the same Section of [11], the Lyapunov drift for the queues of an arbitrary network is bounded above by eq (4.13)

$$\begin{aligned}\Delta\mathbf{X}(\tau) \leq\ & 2B|\mathcal{N}| + 2\sum_{i\in\mathcal{N}} X_i(\tau)\mathrm{E}\{A_i(\tau)|\mathbf{X}(\tau)\} \\ & - 2\mathrm{E}\left\{\sum_{i\in\mathcal{N}} X_i(\tau)\left[\sum_{k\in\mathcal{N}_o^i} F_{i,k}(\tau) - \sum_{k:i\in\mathcal{N}_o^k} F_{k,i}(\tau)\right]|\mathbf{X}(\tau)\right\},\end{aligned} \tag{24}$$

where $F_{i,k}(\tau)$ is the actual service rate of link (i,k) at time slot τ when Algorithm 2 is in use.

For each control $I \in \mathcal{I}$, consider the quantity

$$C(I) = \max_{\mu_i(I) \leq \hat{\mu}_i(I)} \sum_{i \in \mathcal{N}} \left(X_i(\tau) - \sum_{k \in \mathcal{E}_o^i} w_{(ik)}(I) X_k(\tau) \right) \mu_i(I) = \max_{\mu_i(I) \leq \hat{\mu}_i(I)} \sum_{i \in \mathcal{N}} Y_i(\tau, I) \mu_i(I),$$

where $Y_i(\tau, I) = X_i(\tau) - \sum_{k \in \mathcal{E}^i} w_{(ik)}(I) X_k(\tau)$. Note now that for a given control, the above maximization is attained by

$$\mu_i^\star(I) = \begin{cases} \hat{\mu}_i(I) & \text{if } Y_i(\tau, I) > 0 \\ 0 & \text{if } Y_i(\tau, I) \leq 0, \end{cases}$$

i.e. transmitting with full rate from nodes with with positive $Y_i(\tau, I)$ and transmitting zero from nodes with negative $Y_i(\tau, I)$. This explains why the algorithm makes use of the function $(.)^+ \equiv \max[., 0]$. Finally, note that Algorithm 2 solves a second maximization problem at each slot by selecting the control

$$I_\tau^* = \arg\max_{I \in \mathcal{I}} C(I).$$

Consider now any point in the interior of the throughput region, denoted by $\breve{\boldsymbol{\lambda}}$. There exist flow variables $\mathbf{f} \in \mathcal{C}$ for which we will have for each node $\breve{\lambda}_i + \epsilon \leq \sum_{k \in \mathcal{E}_o^i} f_{ik} - \sum_{k : i \in \mathcal{E}_o^k} f_{ki}$. Also, we have for any τ

$$\begin{aligned}
\sum_{i \in \mathcal{N}} X_i(\tau) \left[\sum_{k \in \mathcal{E}_o^i} f_{ik} - \sum_{k : i \in \mathcal{E}_o^k} f_{ki} \right] &= \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{E}_o^i} \left(X_i(\tau) - X_k(\tau) \right) f_{i,k} \\
&= \sum_{I \in \mathcal{I}} \phi_I \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{E}_o^i} \left(X_i(\tau) - X_k(\tau) \right) w_{(i,k)}(I) \mu_i(I) \\
&= \sum_{I \in \mathcal{I}} \phi_I \sum_{i \in \mathcal{N}} \left(X_i(\tau) - \sum_{k \in \mathcal{E}_o^i} X_k(\tau) w_{(i,k)}(I) \right) \mu_i(I) \\
&\leq \sum_{I \in \mathcal{I}} \phi_I C(I) \leq \sum_{I \in \mathcal{I}} \phi_I C(I_\tau^*) \\
&= \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{E}_o^i} \left(X_i(\tau) - X_k(\tau) \right) w_{(i,k)}(I_\tau^*) \mu_i^\star(I_\tau^*) \\
&= \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{E}_o^i} \left(X_i(\tau) - X_k(\tau) \right) \mathrm{E}\left\{ F_{i,k}(\tau, I_\tau^*) | \mathbf{X}(\tau) \right\},
\end{aligned}$$

where I_τ^* is the control selected by Algorithm 2 at time slot τ and the equality below the second inequality follows from the fact that Algorithm 2 is designed exactly to maximize this term at each time slot. Now we can elaborate (24):

$$\begin{aligned}
\Delta\mathbf{X}(\tau) &\leq 2B|\mathcal{N}| + 2\sum_{i\in\mathcal{N}} X_i(\tau)\lambda_i - 2\sum_{i\in\mathcal{N}} X_i(\tau)\mathrm{E}\left\{\sum_{k\in\mathcal{N}_o^i} F_{i,k}(\tau) - \sum_{k:i\in\mathcal{N}_o^k} F_{k,i}(\tau)|\mathbf{X}(\tau)\right\} \\
&= 2B|\mathcal{N}| + 2\sum_{i\in\mathcal{N}} X_i(\tau)\lambda_i - 2\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{E}_o^i} \left(X_i(\tau) - X_k(\tau)\right)\mathrm{E}\left\{F_{i,k}(\tau, I_\tau^*)|\mathbf{X}(\tau)\right\} \\
&\leq 2B|\mathcal{N}| + 2\sum_{i\in\mathcal{N}} X_i(\tau)\lambda_i - 2\sum_{i\in\mathcal{N}} X_i(\tau)\left[\sum_{k\in\mathcal{E}_o^i} f_{ik} - \sum_{k:i\in\mathcal{E}_o^k} f_{ki}\right] \\
&= 2B|\mathcal{N}| - 2\sum_{i\in\mathcal{N}} X_i(\tau)\left[\sum_{k\in\mathcal{E}_o^i} f_{ik} - \sum_{k:i\in\mathcal{E}_o^k} f_{ki} - \lambda_i\right] \leq 2B|\mathcal{N}| - 2\epsilon\sum_{i\in\mathcal{N}} X_i(\tau),
\end{aligned}$$

where the second inequality comes from the aforementioned derivation and the third inequality comes from the fact that the point $\breve{\boldsymbol{\lambda}}$ is selected in the interior of the throughput region. Thus, we have shown that whenever the backlogs are large enough, Algorithm 1 guarantees a negative drift which brings the system to stability as long as the vector $\boldsymbol{\lambda}$ is in the interior of the throughput region. □

Appendix III - More controls

In this subsection we examine closer the control that sends encoded packets (results of XOR operations of packet pairs belonging to two queues). Under this control, if queues a and b are selected, m packet pairs from the queues are transmitted in a slot by using "dummy" packets if necessary to form these pairs. In this case an inefficiency seems to arise as indicated by the following example. Suppose that the control selects to perform XOR operation of 10 packets, queue a has 20 queued packets and queue b 3 packets. Then according to the control specified, 10 packet pairs will be XORed by using 7 "dummy" packets from queue b. However, it would have been more efficient to send 3 pairs of XORed packets, without using any dummy packets, and then transmit 7 uncoded packets from queue a. Since this type of control is not included in the controls specified in the previous Sections, the question arises whether one can do better by introducing more detailed controls. Below we show that this is not the case.

Let us extend the available controls by adding the following ones. If it is decided to transmit packets belonging to both of the queues a and b, then a control $I(a, b, l_o, l_a, l_b)$ may be selected, where l_0, l_a, l_b are nonnegative integers with the following interpretation. Let $r_a \geq r_b$. Then at most l_0 XORed packets may be transmitted and at most l_x, $x = a$ or b uncoded packets may be transmitted from each of the queues. The $l_0 + l_b$ packets must be seen by user b and hence they need to be transmitted at rate r_b. On the other hand the l_a packets need to be seen only by user a and hence they can be transmitted at the higher rate r_a. Since all these packets must be transmitted within a time slot, it must hold

$$\frac{l_0 + l_b}{r_b} + \frac{l_a}{r_a} \leq 1 \tag{25}$$

This type of controls covers the case described in the example above. The controls corresponding to transmission from one of the queues remain the same.

The model used in the previous Sections can be extended to cover the case when this extended set of controls is chosen. The resulting stabilizing policy in this case is similar to the one described in Section 6. Specifically, at time t a reward $C(I)$ is specified for each control, the reward depending on queue sizes at time t, and then the control whose reward is maximized is selected for slot $(t, t+1]$. The reward for a control $I(a, b, l_o, l_a, l_b)$ is given by

$$C(I) = \left(\sum_{i \in \{a,b\}} \left(X_i(t) - \sum_{k \in \mathcal{E}_o^i} X_k(t) w_{(i,k)}^l(I) \right)^+ \right) l_0 + X_a(t) l_a + X_b(t) l_b. \quad (26)$$

Let now c_0, c_a, c_b be the coefficients multiplying l_0, l_a, l_b in (26). Consider all controls $I(a, b, l_o, l_a, l_b)$ where a, b are fixed. If $c_0 = \max\{c_0, c_a, c_b\}$ then among all these controls those that set $l_b = 0$ dominate. For the latter class of controls the reward becomes

$$C(I) = c_0 l_0 + c_a l_a.$$

Taking into account (25) we have

$$C(I) \leq \left(c_0 - c_a \frac{r_a}{r_b} \right) l_0 + c_a r_a.$$

Hence, if

$$c_0 > c_a \frac{r_a}{r_b},$$

then the control with $l_0 = r_b$, $l_a = l_b = 0$ dominates. If on the other hand $c_0 \leq c_a r_a / r_b$ then the control $l_0 = 0$, $l_a = r_a$, $l_b = 0$ dominates. In either case we obtain one of the admissible controls of the policy defined in Section 6. In a similar fashion it can be shown that the cases $c_b = \max\{c_0, c_a, c_b\}$ or $c_a = \max\{c_0, c_a, c_b\}$ result in one of the admissible controls of the policy defined in Section 6.

We see from the discussion above, that under the new extended set of controls, the policy specified in Section 6 will still be optimal.

Appendix IV - Beyond pairwise XOR

To provide some intuition as to why extensions are needed for the case where $|\mathcal{P}| > 2$, we briefly explain here an example with three combined packets. If only partial state feedback is given, the virtual network that captures all state transitions might contain infinite number of nodes. Consider three packets from different flows which are combined together. Then assume that the scheduler receives a NACK from destination 1 and 2 and an ACK from 3. This means that both destination nodes 1,(2) did not correctly decode the packet, but this could be either because packet 2 (1) was not correctly overheard or packet 3 was not correctly overheard. To capture both cases, none of the packets can be characterized as bad; instead, the relay can now estimate new overhearing probabilities. A new state is required to capture this new partial knowledge of the scheduler thus obtained. Repeating this process, we see that in order to capture all the partial knowledge that the relay may have in the construction of the virtual network, we need to introduce infinite number of nodes. This introduces new technical challenges and extensions to the approach used in this paper.