Robust and Optimal Opportunistic Scheduling for Downlink 2-Flow Inter-session Network Coding with Varying Channel Quality

Wei-Cheng Kuo, Chih-Chun Wang
{wkuo, chihw}@purdue.edu
School of Electrical and Computer Engineering, Purdue University, USA

***Abstract*—This paper considers the downlink traffic from a base station to two different clients. Assuming infinite backlog, it is known that *inter-session* network coding (INC) can significantly increase the throughput of each flow. However, the corresponding scheduling solution (assuming dynamic arrivals and requiring bounded delay) is still nascent.**

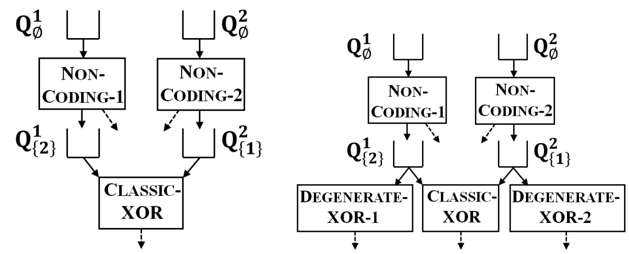
For the 2-flow downlink scenario, we propose the first opportunistic INC + scheduling solution that is provably optimal for time-varying channels, i.e., the corresponding stability region matches the optimal linear-INC capacity. To that end, we first introduce a new *binary INC* operation, which is distinctly different from the traditional wisdom of XORing two overheard packets. We then develop a *queue-length-based* scheduling scheme, which, with the help of the new INC operation, can robustly and optimally adapt to time-varying channel quality. A byproduct of our results is a scheduling scheme for stochastic processing networks (SPNs) with *random departure*. The new SPN results relax the previous assumption of *deterministic departure*, a major limitation of the existing SPN model, by considering stochastic packet departure behavior, and could further broaden the applications of SPN scheduling to other real-world scenarios.

I. Introduction

Since 2000, NC has emerged as a promising technique in communication networks. The seminal work by [1] shows linear intra-session NC achieves the min-cut/max-flow capacity of single-session multi-cast networks. The natural connection of intra-session NC and the *maximum flow* allows the use of back-pressure (BP) algorithms to stabilize intra-session NC traffic, see [2] and the references therein.

However, when there are multiple coexisting sessions, the benefits of *inter-session* network coding (INC) are far from fully utilized. The COPE architecture [3] demonstrated that a simple INC scheme can provide 40%–200% throughput improvement when compared to the existing TCP/IP architecture in a testbed environment. Several analytical attempts have been made to characterize the INC capacity (or provably achievable throughput) for various small network topologies [4]–[7].

However, unlike the case of intra-session NC, there is no direct analogy from INC to the commodity flow. As a result, it is much more challenging to derive BP-based scheduling for INC traffic. We use the following example to illustrate this point. Consider a single source s and two destinations d_1 and d_2. Source s would like to send to d_1 the X_i packets, $i = 1, 2, \cdots$; and send to d_2 the Y_j packets, $j = 1, 2, \cdots$. The simplest INC scheme consists of three operations. OP1: Send uncodedly those X_i that have not been heard by any of $\{d_1, d_2\}$. OP2: Send uncodedly those Y_j that have not been heard by any of $\{d_1, d_2\}$. OP3: Send a linear sum $[X_i + Y_j]$ where X_i has been overheard by d_2 but not by d_1 and Y_j has been overheard by d_1 but not by d_2. For future reference, we denote OP1 to OP3 by NON-CODING-1, NON-CODING-2, and CLASSIC-XOR, respectively.



(a) INC using only 3 operations (b) INC using only 5 operations

Fig. 1. The virtual networks of two INC schemes.

OP1 to OP3 can also be represented by the virtual network (vr-network) in Fig. 1(a). Namely, any newly arrived X_i and Y_j virtual packets[1] (vr-packets) that have not been heard by any of $\{d_1, d_2\}$ are stored in queues $Q^1_\emptyset$ and $Q^2_\emptyset$, respectively. The superscript $k \in \{1, 2\}$ indicates that the queue is for the session-k packets. The subscript $\emptyset$ indicates that those packets have not been heard by any of $\{d_1, d_2\}$. NON-CODING-1 then takes one X_i vr-packet from $Q^1_\emptyset$ and send it uncodedly. If such X_i is heard by d_1, then the vr-packet leaves the vr-network, which is described by the dotted arrow emanating from the NON-CODING-1 block. If X_i is overheard by d_2 but not d_1, then we place it in queue $Q^1_{\{2\}}$, the queue for the overheard session-1 packets. NON-CODING-2 in Fig. 1(a) can be interpreted similarly. CLASSIC-XOR operation takes an X_i from $Q^1_{\{2\}}$ and a Y_j from $Q^2_{\{1\}}$ and sends $[X_i + Y_j]$. If d_1 receives $[X_i + Y_j]$, then X_i is removed from $Q^1_{\{2\}}$ and leaves the vr-network. If d_2 receives $[X_i + Y_j]$, then Y_j is removed from $Q^2_{\{1\}}$ and leaves the vr-network. The transition probability (of the edges) of the vr-network can be computed by analyzing the corresponding random events when transmitting the packet physically.

It is known [8] that with dynamic packet arrivals, any INC

[1] We often use "virtual packets" to refer to the packets (jobs) inside the vr-network.

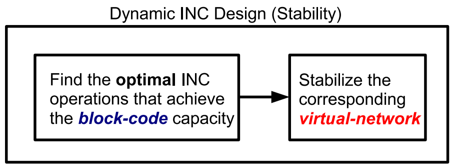


Fig. 2. The two components of optimal dynamic INC design.

scheme that (i) uses only these three operations and (ii) attains bounded decoding delay with arrival rates (R_1, R_2) can always be converted to a scheduling solution that stabilizes the vr-network with arrival rates (R_1, R_2), and vice versa. The INC design problem is thus converted to a scheduling problem on the vr-network. To distinguish the above INC design for dynamical arrivals (the concept of the stability region) and the INC design assuming infinite backlog and decoding delay (the concept of the Shannon capacity), we term the former *the dynamic INC* design problem and the latter *the block-code INC* design problem.

The above vr-network representation also allows us to divide the optimal dynamic INC design problem into solving two major challenges: **Challenge 1**: The example in Fig. 1(a) focuses on dynamic INC schemes using only 3 possible operations. Obviously, the more INC operations one can choose from, the larger the degree of design freedom, and the higher the achievable throughput. *The goal is thus to find a (small) finite set of INC operations that can provably maximize the "block-code" achievable throughput.* **Challenge 2**: Suppose that we have found a set of INC operations that achieves the block-code capacity. However, it does not mean that such a set of INC operations always leads to a dynamic INC design since we still need to consider the delay/stability requirements. Specifically, once the INC operation set is decided, we can derive the corresponding vr-network. *The goal is then to devise a stabilizing scheduling policy for the vr-network, which leads to an equivalent representation of the optimal dynamic INC solution.* See Fig. 2 for the illustration of these two tasks.

Both tasks turn out to be highly non-trivial and optimal dynamic INC solution [4], [8], [9] has been designed only for the scenario of fixed channel quality. Specifically, [10] answers Challenge 1 and shows that for fixed channel quality, the 3 INC operations in Fig. 1(a) plus 2 additional DEGENERATE-XOR operations, see Fig. 1(b) and Section II-B1, can achieve the block-code INC capacity. One difficulty of resolving Challenge 2 is that an INC operation may involve multiple queues simultaneously, e.g., CLASSIC-XOR can only be scheduled when both $Q^1_{\{2\}}$ and $Q^2_{\{1\}}$ are non-empty. This is in sharp contrast with the traditional BP solutions in which each queue can act independently.[2] For the vr-network in Fig. 1(b), [4] circumvents this problem by designing a fixed priority rule that gives strict precedence to the CLASSIC-XOR operation. Alternatively, [8] derives a BP scheduling scheme by noticing that the vr-network in Fig. 1(b) can be decoupled into two vr-subnetworks (one for each data session) so that the queues in each of the vr-subnetworks can be activated independently and the traditional BP results follow.

However, the channel quality varies over time for practical wireless downlink scenarios. Therefore, one should opportunistically choose the most favorable users as receivers, the so-called *opportunistic scheduling* technique. Nonetheless, recently [12] shows that when allowing opportunistic coding+scheduling for time-varying channels, the 5 operations in Fig. 1(b) no longer achieve the block-code capacity. The existing dynamic INC design in [4], [8] are thus strictly suboptimal for time-varying channels since they are based on a suboptimal set of INC operations (recall Fig. 2).

In this work, we propose a new optimal dynamic INC design for 2-flow downlink traffic with time-varying channels. Our detailed contributions are summarized as follows.

Contribution 1: We introduce a new INC operation such that (i) The underlying concept is distinctly different from the traditional wisdom of XORing two overheard packets; (ii) It uses only the ultra-low-complexity binary XOR operation; and (iii) The new INC operation is guaranteed to achieve the best possible capacity of any linear block-code INC solutions.

Contribution 2: The introduction of new INC operations leads to a new vr-network that is different from Fig. 1(b) and for which the existing "*vr-network decoupling + BP*" approach in [8] no longer holds. To answer Challenge 2 of the optimal dynamic INC design, we generalize the results of Stochastic Processing Networks (SPNs) [13], [14] and successfully apply it to the new vr-network. The end result is an opportunistic, dynamic INC solution that is completely *queue-length-based* and can robustly adapt to time-varying channels while achieving the largest possible stability region.

Contribution 3: A byproduct of our results is a scheduling scheme for SPNs with *random departure*. The new results relax the previous assumption of *deterministic departure*, a major limitation of the existing SPN model, by considering stochastic packet departure behavior, and thus could further broaden the applications of SPN scheduling to other real-world scenarios.

The rest of the paper is organized as follows. Section II discusses the existing results on INC design and on SPN scheduling. Sections III and IV propose a new INC operation and a new SPN scheduling solution, respectively. Section V elaborates how to combine the new INC operation and the new SPN scheduling to derive the optimal dynamic INC solution. Section VI contains the simulation results and Section VII concludes the paper.

II. PROBLEM FORMULATION AND EXISTING RESULTS

In this section, we will introduce the problem formulation and then discuss the latest results on the block-code LNC literature (related to Challenge 1) and on the SPN scheduling work (related to solving Challenge 2).

[2]To be more precise, a critical assumption in [C.1 [11]] is that if two queues Q_1 and Q_2 can be *activated* at the same time, then we can also choose to activate exactly one of the queues if desired. This is unfortunately not the case in the vr-network. E.g., CLASSIC-XOR activates both $Q^1_{\{2\}}$ and $Q^2_{\{1\}}$ but no coding operation in Fig. 1(a) activates only one of $Q^1_{\{2\}}$ and $Q^2_{\{1\}}$.

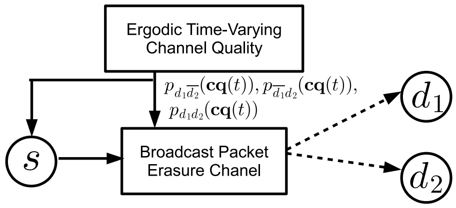


Fig. 3. The time-varying broadcast packet erasure channel.

A. Problem Formulation — The Broadcast Erasure Channel

We model the 1-base-station/2-client downlink traffic as a broadcast packet erasure channel. See Fig. 3 for illustration. The detailed model description is as follows. Consider the following slotted transmission system.

Dynamic Arrival: In the beginning of every time t, there are $A_1(t)$ session-1 packets and $A_2(t)$ session-2 packets arriving at source s. We assume that $A_1(t)$ and $A_2(t)$ are i.i.d. integer-valued random variables with mean $(\mathsf{E}\{A_1(t)\}, \mathsf{E}\{A_2(t)\}) = (R_1, R_2)$ and bounded support. Recall that X_i and Y_j, $i, j \in \mathbb{N}$, denote the session-1 and session-2 packets, respectively.

Time-Varying Channel: We model the time-varying channel quality by a random process $\mathsf{cq}(t)$, which, as will be elaborated shortly after, decides the reception probability of the *broadcast packet erasure channel.* We consider two types of random processes: $\mathsf{cq}(t)$ being i.i.d. or being periodic. Let CQ denote the support of $\mathsf{cq}(t)$ and we assume $|\mathsf{CQ}|$ is finite. For any $a \in \mathsf{CQ}$, we use f_c to denote the expected/long-term frequency of $\mathsf{cq}(t) = c$. Obviously $\sum_{c\in\mathsf{CQ}} f_a = 1$ since the total frequency is 1.

Broadcast Packet Erasure Channel: For each time slot t, source s can transmit one packet, which will be received by a random subset of destinations $\{d_1, d_2\}$. Specifically, there are 4 possible *reception status* $\{\overline{d_1 d_2}, d_1\overline{d_2}, \overline{d_1}d_2, d_1d_2\}$, e.g., the reception status $\mathsf{rcpt} = d_1\overline{d_2}$ means that the packet is received by d_1 but not d_2. The reception status probabilities can be described jointly by a vector $\vec{p} \triangleq (p_{\overline{d_1 d_2}}, p_{d_1\overline{d_2}}, p_{\overline{d_1}d_2}, p_{d_1d_2})$. For example, $\vec{p} = (0, 0.5, 0.5, 0)$ means that every time we transmit a packet, with 0.5 probability it will be received by d_1 only and with 0.5 probability it will be received by d_2 only. It will never be received by d_1 and d_2 simultaneously. In contrast, if we have $\vec{p} = (0, 0, 0, 1)$, then it means that the packet is always received by d_1 and d_2 simultaneously.

Opportunistic INC: Since the reception probability is decided by the channel quality, we write $\vec{p}(\mathsf{cq}(t))$ as a function of $\mathsf{cq}(t)$ at time t. In the beginning of time t, we assume that s is aware of the channel quality $\mathsf{cq}(t)$ (and thus knows $\vec{p}(\mathsf{cq}(t))$) so that s can opportunistically decide how to encode the packet for the current time slot. See Fig. 3.

ACKnowledgement: In the end of time t, both d_1 and d_2 will report back to s whether they have received the transmitted packet or not. This models the use of ACK.

B. Existing Results on Block INC Design

[12] focuses on the above setting but considers the infinite backlog block-code design instead of dynamic arrivals. Two findings of [12] are summarized here.

1) The 5 INC operations in Fig. 1(b) are no longer optimal for time-varying channels: In Section I, we have detailed 3 INC operations: NON-CODING-1, NON-CODING-2, and CLASSIC-XOR. Two additional INC operations are introduced in [10]: DEGENERATE-XOR-1 and DEGENERATE-XOR-2 as illustrated in Fig. 1(b). Specifically, DEGENERATE-XOR-1 is designed to handle the degenerate case in which $Q^1_{\{2\}}$ is non empty but $Q^2_{\{1\}} = \emptyset$. Namely, there is at least one X_i packet overheard by d_2 but there is no Y_j packet overheard by d_1. Not having such Y_j implies that one cannot send $[X_i + Y_j]$ (the CLASSIC-XOR operation). An alternative is thus to send the overheard X_i uncodedly (as if sending $[X_i + 0]$). We term this operation DEGENERATE-XOR-1. One can see from Fig. 1(b) that DEGENERATE-XOR-1 takes a vr-packet from $Q^1_{\{2\}}$ as input. If d_1 receives it, the vr-packet will leave the vr-network. DEGENERATE-XOR-2 is the symmetric version of DEGENERATE-XOR-1.

We use the following example to illustrate the sub-optimality of the above 5 operations. Suppose s has an X packet for d_1 and a Y packet for d_2 and consider a duration of 2 time slots. Also suppose that s knows beforehand that the time-varying channel will have (i) $\vec{p} = (0, 0.5, 0.5, 0)$ for slot 1; and (ii) $\vec{p} = (0, 0, 0, 1)$ for slot 2. The goal is to transmit as many packets in 2 time slots as possible.

Solution 1: INC based on the 5 operations in Fig. 1(b). In the beginning of time 1, both $Q^1_{\{2\}}$ and $Q^2_{\{1\}}$ are empty. Therefore, we can only choose either NON-CODING-1 or NON-CODING-2. Since the setting is symmetric, without loss of generality we assume that we choose NON-CODING-1 and thus send X uncodedly. Since $\vec{p} = (0, 0.5, 0.5, 0)$ in slot 1, there are only two cases to consider. Case 1: X is received only by d_1. In this case, we can send Y in the second time slot, which is guaranteed to arrive at d_2 since $\vec{p} = (0, 0, 0, 1)$ in slot 2. The total sum rate is sending 2 packets (X and Y) in 2 time slots. Case 2: X is received only by d_2. In this case, $Q^1_{\{2\}}$ contains one packet X, and $Q^2_{\emptyset}$ contains one packet Y, and all the other queues in Fig. 1(b) are empty. We can thus choose either NON-CODING-2 or DEGENERATE-XOR-1 for slot 2. Slot 2 will then deliver 1 packet to either d_2 or d_1, depending on which INC operation we choose. Since no packet is delivered in slot 1, the total sum rate is 1 packet in 2 time slots. Since both cases have probability 0.5, the expected sum rate is $2 \cdot 0.5 + 1 \cdot 0.5 = 1.5$ packets in 2 time slots.

An optimal solution: We can achieve strictly better throughput by introducing new INC operations. Specifically, in slot 1, we send the linear sum $[X + Y]$ *even though neither X nor Y has ever been transmitted*, a distinct departure from the existing 5-operation-based solutions.

Again consider two cases: Case 1: $[X+Y]$ is received only by d_1. In this case, we let s send Y uncodedly in slot 2. Since $\vec{p} = (0, 0, 0, 1)$ in slot 2, the packet Y will be received by both d_1 and d_2. d_2 is happy since it has now received the desired Y packet. d_1 can use Y together with the $[X + Y]$ packet received in slot 1 to decode its desired X packet. Therefore, we deliver 2 packets (X and Y) in 2 time slots. Case 2: $[X+Y]$ is received only by d_2. In this case, we let[3] s sends X uncodedly in slot 2. By the symmetric argument of Case 1, we deliver 2 packets (X and Y) in 2 time slots. As a result, the sum-rate of the new solution is 2 packets in 2 slots, a 33% improvement over the existing solution.

Remark: This example focuses on a 2-time-slot duration due to the simplicity of the analysis. It is worth noting that the throughput improvement persists even for infinitely many time slots. See the simulations results in Section VI.

2) The block-code capacity region for linear INC: We summarize the high-level description of [Proposition 1, [12]]:

Proposition 1: [Proposition 1, [12]] For the block-code setting, a rate vector (R_1, R_2) can be achieved by a linear INC scheme if and only if a specifically constructed linear programming (LP) problem is feasible. Given any (R_1, R_2), the LP problem of interest involves $18 \cdot |\mathsf{CQ}| + 7$ non-negative variables and $|\mathsf{CQ}| + 17$ (in-)equalities and can be explicitly computed.

Our goal is to design a dynamic INC scheme, of which the stability region matches the block-code capacity region. To that end, we will later show that the stability region of our dynamic INC matches the region described in Proposition 1.

C. Stochastic Processing Networks (SPNs)

The main tool that we use to stabilize the vr-network is scheduling for the stochastic processing networks (SPNs). In the following, we will discuss the existing results on SPNs.

1) The Main Feature of SPNs: The SPN is a generalization of the store-and-forward networks. In an SPN, a packet can not be transmitted directly from one queue to another queue through links. Instead, it must first be processed by a unit called "Service Activity" (SA). The SA first collects a certain amount of packets from one or more queues (named the *input queues*), jointly processes these packets, generates a new set of packets, and finally redistributes them to another set of queues (named the *output queues*). There is one rule for the SA activation: *An SA can be activated only when all its input queues can provide enough amount of packets for the SA to process.* This SA rule captures directly the INC behavior. Other applications of SPNs include the video streaming problem [15] and the Map-&-Reduce scheduling [16].

2) SPNs with Deterministic Departure: All the existing SPN scheduling solutions [13], [14] assume a special class of SPNs, which we call SPNs with deterministic departure. We elaborate the detailed definition in the following.

Consider a time-slotted system with i.i.d./periodic channel quality $\mathsf{cq}(t)$. An SPN consists of three components: the input activities (IAs), the service activities (SAs), and the queues. We suppose that there are K queues, M IAs, and N SAs in the SPN.

Input Activities: Each IA denotes a session (or a flow) of packets and outputs packets to a deterministic set of queues when activated. When an IA m is activated, it sends $\alpha_{k,m}$ packets to queue k. Let $\mathcal{A} \in \mathbb{R}^{K*M}$ be the "input matrix" with $\mathcal{A}_{k,m} = -\alpha_{k,m}$, for all m and k. At each time t, a random subset of IAs will be activated. Equivalently, we define $\mathbf{a}(t) \triangleq (a_1(t), a_2(t), \cdots, a_M(t)) \in \{0,1\}^M$ as the "arrival vector" at time t. If $a_m(t) = 1$, then IA m is activated at time t. We assume that the random vector $\mathbf{a}(t)$ is i.i.d. over time with the average rate vector $\mathbf{R} = \mathsf{E}\{\mathbf{a}(t)\}$. We also assume that the $\mathcal{A}$ matrix is a fixed (deterministic) system parameter and all the randomness of IAs lies in $\mathbf{a}(t)$.

Service Activities: When SA n is activated, it takes packets from a set of queues, denoted by $\mathfrak{I}_n$, and sends packets to another set of queues, denoted by $\mathfrak{O}_n$. $\mathfrak{I}_n$ and $\mathfrak{O}_n$ are named the input and output queues of SA n, respectively. Specifically, when SA n is activated, it takes $\beta_{k,n}$ packets from queue k for all $k \in \mathfrak{I}_n$ and send $\beta'_{\tilde{k},n}$ packets to queue $\tilde{k}$ for all $\tilde{k} \in \mathfrak{O}_n$. Let $\mathcal{B} \in \mathbb{R}^{K*N}$ be the "service matrix" with $\mathcal{B}_{k,n} = \beta_{k,n}$ if $k \in \mathfrak{I}_n$ and $\mathcal{B}_{k,n} = -\beta'_{k,n}$ if $k \in \mathfrak{O}_n$. In the beginning of each time t, the SPN scheduler is made aware of the current channel quality $\mathsf{cq}(t)$ and can choose to "activate" a subset of the SAs. Let $\mathbf{x}(t) \in \{0,1\}^N$ be the "service vector" at time t. If $\mathbf{x}_n(t) = 1$, then it implies that we choose to activate SA n at time t. Note that for some applications we may need to impose the condition that some of the SAs cannot be scheduled in the same time slot. To model this *interference constraint*, we require $\mathbf{x}(t)$ to be chosen from a pre-defined set of binary vectors $\mathfrak{X}$. Define Λ to be the convex hull of $\mathfrak{X}$ and let Λ° be the interior of Λ.

Acyclicness and Time-Varying Channels: The input/outuput queues $\mathfrak{I}_n$ and $\mathfrak{O}_n$ of the SAs can be used to plot the corresponding SPN. We assume that the SPN is acyclic. We allow the input/ouput service rates $\beta_{k,n}$ and $\beta'_{k,n}$ to depend on the current channel quality $\mathsf{cq}(t)$, but assume that $\mathsf{cq}(t)$ does not change $\mathfrak{I}_n$ and $\mathfrak{O}_n$, the topology of the SPN. For simplicity, we write $\mathcal{B}$ as a deterministic function $\mathcal{B}(c)$ where $c = \mathsf{cq}(t)$ to highlight the assumption that $\beta_{k,n}$ and $\beta'_{k,n}$ may depend on $\mathsf{cq}(t)$. Recall that f_c is the relative frequency of $\mathsf{cq}(t) = c$. We then have the following proposition.

Definition 1: An arrival rate vector $\mathbf{R}$ is "feasible" if there exist $\mathbf{s}_c \in \Lambda$ for all $c \in \mathsf{CQ}$ such that

$$\mathcal{A} \cdot \mathbf{R} + \sum_{c \in \mathsf{CQ}} f_c \cdot \mathcal{B}(c) \cdot \mathbf{s}_c = 0. \tag{1}$$

A rate vector $\mathbf{R}$ is "strictly feasible" if there exist $\mathbf{s}_c \in \Lambda^\circ$ for all $c \in \mathsf{CQ}$ such that (1) holds.

Proposition 2: [A combination of [13], [14]] Only feasible $\mathbf{R}$ can possibly be stabilized. Moreover, there exists an SPN scheduler that can stabilize all $\mathbf{R}$ that are strictly feasible.

The achievability part for SPNs with deterministic departure (Proposition 2) is proven by the Deficit Max-Weight (DMW) algorithm in [13] and by the Perturb Max-Weight (PMW) algorithm in [14].

3) SPNs with Random Departure: Although the SPN with deterministic departure is relatively well understood, those SPN scheduling results cannot be applied to the INC vr-network. The reason is as follows. When a packet is broadcast

[3]ACK is critical in this scheme. I.e., s needs to know whether it is d_1 or d_2 who has received $[X+Y]$ in slot 1 before deciding whether to send Y or X in slot 2.

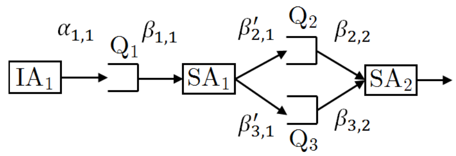


Fig. 4. An SPN with random departure.

by the base station, it can arrive at a random subset of receivers with certain probability distributions. Therefore, the vr-packets move among the vr-queues according to some probability distribution. This is not compatible with the deterministic departure SPN model, in which when an SA is activated we know *deterministically* $\beta_{k,n}(c)$ and $\beta'_{k,n}(c)$, the opportunistic service rates when the channel quality is $\mathsf{cq}(t) = c$. We call the SPN model that allows random $\beta_{k,n}(c)$ and $\beta'_{k,n}(c)$ the SPN with random departure.

SPNs with random departure provide a unique challenge for the scheduling design. [13] provides the following example illustrating this issue. Fig. 4 describes an SPN with 6 transition edges. We assume IA1 is activated at every time slot and $\alpha_{1,1} = \beta_{1,1} = \beta_{2,2} = \beta_{3,2} = 1$. Namely, for every time t, 1 packet will enter Q_1; every time we activate SA1, 1 packet will leave Q_1; every time we activate SA2, 1 packet will leave Q_2 and 1 packet will leave Q_3. These 4 transitions are all deterministic. The two transitions SA1 $\to Q_2$ and SA1 $\to Q_3$ are random. Specifically, we assume that there are two possible values of the pair $(\beta'_{2,1}, \beta'_{3,1})$: $(\beta'_{2,1}, \beta'_{3,1}) = (1, 0)$ with probability 0.5 and $(\beta'_{2,1}, \beta'_{3,1}) = (0, 1)$ with probability 0.5. That is, whenever SA1 is activated, it takes a packet from Q_1, and with probability 0.5 this packet goes to Q_2. Otherwise, this packet goes to Q_3. The random departure of SA1 implies that the queue length difference $|Q_2| - |Q_3|$ forms a binary random walk. Note that SA2 has no impact on $|Q_2| - |Q_3|$ since it always takes 1 packet from each of the queues. The analysis of the random walk shows that $|Q_2| - |Q_3|$ goes unbounded with rate $\sqrt{t}$. And hence there is no scheduling algorithm which can stabilize both $|Q_2|$ and $|Q_3|$ simultaneously.

4) The Deficit Maximum Weight (DMW) Scheduling: Since our scheme is based on the DMW algorithm, we briefly describe in the following the DMW scheduling.

In DMW algorithm [13], each queue k maintains a real-valued counter $q_k(t)$, called the virtual queue length. Initially, $q_k(t)$ is set to 0. For comparison, the actual queue length is denoted by $Q_k(t)$ instead.

The key feature of a DMW algorithm is that it makes a back-pressured scheduling decision based on the virtual queue-length, not on the actual queue length. Specifically, for each time t, we choose the service vector (scheduling decision) by

$$\mathbf{x}^*(t) = \arg\max_{\mathbf{x}\in\mathfrak{X}} \mathbf{d}^T(t) \cdot \mathbf{x}, \tag{2}$$

where $\mathbf{d}(t)$ is the back pressure vector defined as $\mathbf{d}(t) = \mathcal{B}^T(\mathsf{cq}(t))\mathbf{q}(t)$, $\mathcal{B}(\mathsf{cq}(t))$ is the service matrix $\mathcal{B}$ when the channel quality is $\mathsf{cq}(t)$, and $\mathbf{q}(t)$ is the vector of the virtual queue lengths. We then update $\mathbf{q}(t)$ according to the transition matrices ($\mathcal{A}$ and $\mathcal{B}$) and the flow conservation law:

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathcal{A} \cdot \mathbf{a}(t) - \mathcal{B}(\mathsf{cq}(t)) \cdot \mathbf{x}^*(t). \tag{3}$$

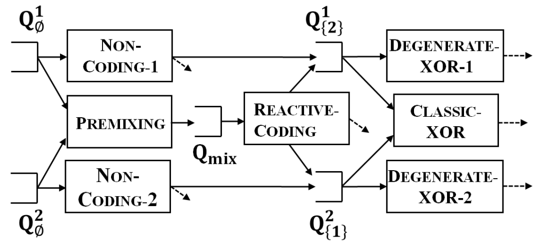


Fig. 5. The virtual network of the proposed new INC solution.

Unlike the actual queue lengths $Q_k(t)$, which is always ≥ 0, the virtual queue length $\mathbf{q}(t)$ can be smaller than 0 when updated via (3). That is, we do not need to take the projection to positive numbers when computing $\mathbf{q}(t)$.

Although the virtual queue length $q_k(t)$ is always updated according to (3). The actual queue length has to follow the SPN rule. That is, suppose SA n has been scheduled according to (2) but for at least one of its input queues, say queue k, the actual queue length $Q_k(t)$ is smaller than $\beta_{k,n}$, the number of packets that are supposed to leave queue k. Then DMW simply skips scheduling SA n for this particular time slot.

III. THE PROPOSED NEW INC SOLUTION

The proposed new INC solution is described as follows. We build upon the existing 5 operations, NON-CODING-1, NON-CODING-2, CLASSIC-XOR, DEGENERATE-XOR-1, and DEGENERATE-XOR-2. See Fig. 1(b) and the discussion in Sections I and II-B1. We add 2 additional operations, termed PREMIXING and REACTIVE-CODING, and 1 additional queue, termed Q_{mix}. We plot the vr-network of the new scheme in Fig. 5. From Fig. 5, we can clearly see that PREMIXING involves both $Q^1_\emptyset$ and $Q^2_\emptyset$ as input and outputs to Q_{mix}. REACTIVE-CODING involves Q_{mix} as input and outputs to $Q^1_{\{2\}}$ or $Q^2_{\{1\}}$ or simply lets the vr-packet leave the vr-network (described by the dotted arrow). For every time instant, we can choose one of the 7 operations and the goal is to stabilize the vr-network. In the following, we describe in details how these two INC operations work and how to integrate them with the other 5 operations. Our description contains 4 parts.

Part I: The two operations, NON-CODING-1 and NON-CODING-2, remain the same.

Part II: We now describe the new operation PREMIXING. We can choose PREMIXING only if both $Q^1_\emptyset$ and $Q^2_\emptyset$ are non-empty. Namely, there are X_i packets and Y_j packets that have not been heard by any of d_1 and d_2. Whenever we schedule PREMIXING, we choose one X_i from $Q^1_\emptyset$ and one Y_j from $Q^2_\emptyset$ and send $[X_i + Y_j]$. If neither d_1 nor d_2 receives it, both X_i and Y_j remain in their original queues.

If at least one of $\{d_1, d_2\}$ receives it, we do the following. We remove *both* X_i and Y_j from their individual queues. We insert a tuple $(\mathsf{rcpt}; X_i, Y_j)$ into Q_{mix}. That is, unlike the other queues for which each entry is a single vr-packet, each entry of Q_{mix} is a tuple.

TABLE I
A SUMMARY OF THE REACTIVE-CODING OPERATION

Current Reception Status $\mathbf{rcpt}(t)$	Departure	Insertion		
		$\mathbf{rcpt}^*{=}d_1\overline{d_2}$ (Send Y_j^*)	$\mathbf{rcpt}^*{=}\overline{d_1}d_2$ (Send X_i^*)	$\mathbf{rcpt}^*{=}d_1d_2$ (Send X_i^*)
$d_1\overline{d_2}$	Remove the tuple from Q_{mix}	Insert Y_j^* to $Q^2_{\{1\}}$	Insert X_i^* to $Q^2_{\{1\}}$	Insert Y_j^* to $Q^2_{\{1\}}$
$\overline{d_1}d_2$		Insert Y_j^* to $Q^1_{\{2\}}$	Insert X_i^* to $Q^1_{\{2\}}$	Insert X_i^* to $Q^1_{\{2\}}$
d_1d_2		Do not insert any packet back into the virtual network.		
$\overline{d_1}\overline{d_2}$	Do nothing			

The first coordinate of $(\mathsf{rcpt}; X_i, Y_j)$ is rcpt, the reception status of $[X_i+Y_j]$. For example, if $[X_i+Y_j]$ was received by d_2 but not by d_1, then we set/record $\mathsf{rcpt} = \overline{d_1}d_2$; If $[X_i+Y_j]$ was received by both d_1 and d_2, then $\mathsf{rcpt} = d_1d_2$. The second and third coordinates store the participating packets X_i and Y_j separately. The reason why we do not store the linear sum directly is due to the new REACTIVE-CODING operation.

Part III: We now describe the new operation REACTIVE-CODING. For any time t, we can choose REACTIVE-CODING only if there is at least one tuple $(\mathsf{rcpt}; X_i, Y_j)$ in Q_{mix}. Choose one tuple from Q_{mix} and denote it by $(\mathsf{rcpt}^*; X_i^*, Y_j^*)$. We now describe the encoding part of REACTIVE-CODING.

Whenever we schedule REACTIVE-CODING, if $\mathsf{rcpt}^* = d_1\overline{d_2}$, send Y_j^*. If $\mathsf{rcpt}^* = \overline{d_1}d_2$, send X_i^*. If $\mathsf{rcpt}^* = d_1d_2$, send X_i^*. One can see that the coding operation depends on the reception status rcpt^* when $[X_i^*+Y_j^*]$ was first transmitted. This is why it is named REACTIVE-CODING.

The movement of the vr-packets depends on the current reception status of time t, denoted by $\mathsf{rcpt}(t)$, and also on the old reception status rcpt^* when the sum $[X_i^*+Y_j^*]$ was originally transmitted. The detailed movement rules are described in Table I. The way to interpret the table is as follows. For example, when $\mathsf{rcpt}(t) = \overline{d_1}\overline{d_2}$, i.e., neither d_1 nor d_2 receives the current transmission, then we do nothing, i.e., keep the tuple inside Q_{mix}. On the other hand, we remove the tuple from Q_{mix} whenever $\mathsf{rcpt}(t) \in \{d_1\overline{d_2}, \overline{d_1}d_2, d_1d_2\}$. If $\mathsf{rcpt}(t) = d_1d_2$, then we remove the tuple but do not insert any vr-packet back to the vr-network, see the second last row of Table I. The tuple essentially leaves the vr-network in this case. If $\mathsf{rcpt}(t) = d_1\overline{d_2}$ and $\mathsf{rcpt}^* = d_1d_2$, then we remove the tuple from Q_{mix} and insert Y_j^* to $Q^2_{\{1\}}$. The rest of the combinations can be read from Table I in the same way. One can verify that the optimal INC example introduced in Section II-B1 is a direct application of the PREMIXING and REACTIVE-CODING operations.

Before we continue describing the slight modification to CLASSIC-XOR, DEGENERATE-XOR-1, and DEGENERATE-XOR-2, we briefly explain why the combination of PREMIXING and REACTIVE-CODING works. To facilitate discussion, we call the time slot in which we use PREMIXING to transmit $[X_i^*+Y_j^*]$ "slot 1" and the time slot in which we use REACTIVE-CODING "slot 2." For example, if $\mathsf{rcpt}^* = d_1\overline{d_2}$ and $\mathsf{rcpt}(t) = d_1d_2$, then it means that d_1 receives $[X_i^*+Y_j^*]$ and Y_j^* in slots 1 and 2, respectively and d_2 receives Y_j^* in slot 2. In this case, d_1 can decode the desired X_i^* and d_2 directly receives the desired Y_j^*. We now consider the perspective of the vr-network. Table I shows that the tuple will be removed from Q_{mix} and leave the vr-network. Therefore, no queue in the vr-network stores any of X_i^* and Y_j^*. This correctly reflects the fact that both X_i^* and Y_j^* have been received by their intended destinations.

Another example is when $\mathsf{rcpt}^* = \overline{d_1}d_2$ and $\mathsf{rcpt}(t) = d_1\overline{d_2}$. In this case, d_2 receives $[X_i^*+Y_j^*]$ in slot 1 and d_1 receives X_i^* in slot 2. From the vr-network's perspective, the movement rule (see Table I) removes the tuple from Q_{mix} and insert an X_i^* packet to $Q^2_{\{1\}}$. Since a vr-packet is removed from a session-1 queue[4] Q_{mix} and inserted to a session-2 queue $Q^2_{\{1\}}$, the total number of vr-packets in the session-1 queue decreases by 1. This correctly reflects the fact that d_1 has received 1 desired packet X_i^* in slot 2.

An astute reader may wonder why we can put X_i^*, a session-1 packet, into a session-2 queue $Q^2_{\{1\}}$. The reason is that whenever d_2 receives X_i^* in the future, it can recover its desired Y_j^* by subtracting X_i^* from the linear sum $[X_i^*+Y_j^*]$ it received in slot 1. Therefore, X_i^* is now information-equivalent to Y_j^*, a session-2 packet. Moreover, d_1 has received X_i^*. Therefore, X_i^* is no different than a session-2 packet that has been overheard by d_1. As a result, it is fit to put X_i^* in $Q^2_{\{1\}}$.

Part IV: We now describe the slight modification to CLASSIC-XOR, DEGENERATE-XOR-1, and DEGENERATE-XOR-2. A unique feature of the new scheme is that some packets in $Q^2_{\{1\}}$ may be a X_i^* packet that is inserted by REACTIVE-CODING when $\mathsf{rcpt}^* = \overline{d_1}d_2$ and $\mathsf{rcpt}(t) = d_1\overline{d_2}$. (Also some $Q^1_{\{2\}}$ packets may be Y_j^*.) However, in our previous discussion, we have shown that those X_i^* in $Q^2_{\{1\}}$ is information-equivalent to a Y_j^* packet overheard by d_1. Therefore, in the CLASSIC-XOR operation, we should not insist on sending $[X_i+Y_j]$ but can also send $[P_1+P_2]$ as long as P_1 is from $Q^1_{\{2\}}$ and P_2 is from $Q^2_{\{1\}}$. The same relaxation must be applied to DEGENERATE-XOR-1 and DEGENERATE-XOR-2 operations. Other than this slight relaxation, the three operations work in the same way as previously described in Sections I and II-B1.

The new two operations PREMIXING and REACTIVE-CODING allow us to achieve the linear block-code capacity for any time-varying channels. We conclude this section by listing in Table II the transition probabilities of half of the edges of the vr-network of Fig. 5. For example, when we schedule PREMIXING, we remove a packet from $Q^1_\emptyset$ if at least one of $\{d_1, d_2\}$ receives it. As a result, the transition probability along the $Q^1_\emptyset \to$PREMIXING edge is $p_{d_1\vee d_2} \triangleq p_{d_1\overline{d_2}} + p_{\overline{d_1}d_2} + p_{d_1d_2}$. All the other transition probabilities in Table II can be derived similarly. The transition probability of the other half of the edges can be derived by symmetry.

IV. THE PROPOSED SCHEDULING SOLUTION

In this section, we first formalize the model of SPNs with random departure and then we propose a new scheme that

[4] Q_{mix} is regarded as both a session-1 and a session-2 queue.

TABLE II
A SUMMARY OF THE TRANSITION PROBABILITY OF THE VIRTUAL NETWORK IN FIG. 5, WHERE $p_{d_1\vee d_2} \triangleq p_{d_1\overline{d_2}} + p_{\overline{d_1}d_2} + p_{d_1d_2}$; $p_{d_1} \triangleq p_{d_1\overline{d_2}} + p_{d_1d_2}$; NC1 STANDS FOR NON-CODING-1; CX STANDS FOR CLASSIC-XOR; DX1 STANDS FOR DEGENERATE-XOR-1; PM STANDS FOR PREMIXING; RC STANDS FOR REACTIVE-CODING.

Edge	Trans. Prob.	Edge	Trans. Prob.
$Q^1_\emptyset \to$NC1	$p_{d_1\vee d_2}$	$Q^1_\emptyset \to$PM	$p_{d_1\vee d_2}$
NC1$\to Q^1_{\{2\}}$	$p_{\overline{d_1}d_2}$	PM$\to Q_{\text{mix}}$	$p_{d_1\vee d_2}$
$Q^1_{\{2\}} \to$DX1	p_{d_1}	$Q_{\text{mix}} \to$RC	$p_{d_1\vee d_2}$
$Q^1_{\{2\}} \to$CX	p_{d_1}	RC$\to Q^1_{\{2\}}$	$p_{\overline{d_1}d_2}$

achieves the optimal throughput region for SPNs with random departure. We conclude this section by providing the key steps of the corresponding stability/throughput analysis.

A. A Simple SPN model with Random Departure

Although our solution applies to general SPNs with random departure, for illustration purposes we describe our scheme by focusing on a simple SPN model with random departure, which we termed the (0,1) random SPN. The (0,1) random SPN includes the INC vr-network in Section III as a special example and is thus sufficient for our discussion.

Recall the definitions in Section II-C2 for SPNs with deterministic departure (we use deterministic SPNs as shorthand). The differences between the (0,1) random SPN and the deterministic SPN are:

Difference 1: In a deterministic SPN, SA n can be scheduled only if for all k in the input queues $\mathfrak{I}_n$, queue k has at least $\beta_{k,n}$ number of packets in the queue. For comparison, In a (0,1) random SPN, SA n can be scheduled only if for all $k \in \mathfrak{I}_n$, queue k has at least 1 packet in the queue.

Difference 2: In a deterministic SPN, when SA n is scheduled, for all $k \in \mathfrak{I}_n$, exactly $\beta_{k,n}$ number of packets will leave queue k. In a (0,1) deterministic SPN, when SA n is scheduled, for all $k \in \mathfrak{I}_n$, the number of packets leaving queue k is a binary random variable with mean $\overline{\beta_{k,n}}$. Namely, with probability $\overline{\beta_{k,n}}$, 1 packet will leave queue k and with probability $1-\overline{\beta_{k,n}}$ no packet will leave queue k.

Difference 3: In a (0,1) deterministic SPN, when SA n is scheduled, for all $k \in \mathfrak{O}_n$, the number of packets entering queue k is a binary random variable with mean $\overline{\beta'_{k,n}}$.

One can easily verify that the INC vr-networks in Figs. 1(a), 1(b), and 5 are special examples of the (0,1) random SPN.

B. The Proposed Solution For (0,1) Random SPNs

For any constant $\epsilon > 0$, say $\epsilon = 0.001$, the proposed scheme works as follows.

Similar to the DMW algorithm, each queue k maintains a real-valued counter $q_k(t)$, the virtual queue length. Initially, $q_k(t)$ is set to 0. For any time t, each entry of the actual service matrix $\mathcal{B}$ takes values in either 0 or 1 since we are focusing on a (0,1) random SPN. We compute $\overline{\mathcal{B}(\mathsf{cq}(t))} \triangleq \mathsf{E}(\mathcal{B}|\mathsf{cq}(t))$, the *average* service matrix. We then slightly augment the average input service rates $\overline{\beta_{k,n}}$ in $\overline{\mathcal{B}(\mathsf{cq}(t))}$ (those rates from queue k to SA n) by $\overline{\beta_{k,n}}^{\epsilon} = \epsilon\cdot(1-\overline{\beta_{k,n}}) + \overline{\beta_{k,n}}$. The new average service rate matrix is denoted by $\overline{\mathcal{B}(\mathsf{cq}(t))}^{\epsilon}$.

Then for each time t, we choose the service vector by the back-pressure decision rule (2) except for that the back-pressure vector $\mathbf{d}(t)$ is now computed by

$$\mathbf{d}(t) = \left(\overline{\mathcal{B}(\mathsf{cq}(t))}^{\epsilon}\right)^T \mathbf{q}(t). \tag{4}$$

That is, we use the augmented average service matrix $\overline{\mathcal{B}(\mathsf{cq}(t))}^{\epsilon}$. We then update $\mathbf{q}(t)$ by

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathcal{A}\cdot\mathbf{a}(t) - \overline{\mathcal{B}(\mathsf{cq}(t))}^{\epsilon}\cdot\mathbf{x}^*(t). \tag{5}$$

In short, we borrow the idea of DMW so that we can make scheduling decisions based on the virtual queue lengths $q_k(t)$ that can take negative values. But then we update $q_k(t)$ only by the average service rates rather than the actual service rates. Finally, we add the ϵ-augmentation but *only* to the average input service rates $\overline{\beta_{k,n}}$, not the output service rates $\overline{\beta'_{k,n}}$.

The actual queue lengths $Q_k(t)$ are updated based on what actually happens in the SPN. For example, suppose the above $q_k(t)$-based rule prompts us to schedule SA n but queue k is empty $Q_k(t) = 0$ for at least one $k \in \mathfrak{I}_n$. By the SPN rule, we cannot schedule such SA n and we will simply skip this SA. Therefore, even when the virtual queue $q_k(t+1)$ is updated by (5), the actual queue length $Q_k(t+1) = Q_k(t)$ remains unchanged since SA n is skipped in the actual SPN.

C. Performance Analysis

The example in Section II-C3 shows that one challenge of the SPN with random departure is that $Q_k(t)$ may grow sublinearly when the deterministic SPN can still be stabilized. However, from a throughput perspective, sublinear growth means that the throughput penalty incurred by the growing queues is negligible since the throughput is the average number of the packet arrivals per second. Moreover, for any scheme A that achieves sublinearly growing queues, we can often convert it to a bounded queue scheme by (i) Run scheme A until any of the sublinearly growing queue length hits some pre-defined threshold; (ii) Stop scheme A and run a naive scheme B that focuses on "draining" the queues of the network; (iii) When running scheme B, put any any new arrival packets into a separate buffer Q; (iv) After scheme B successfully drains out all the queues, we start to run scheme A again and we inject the packets collected in Q gradually back to the system. The above 4 steps loose some throughput optimality but can be made arbitrarily close to optimal when choosing a large threshold in Step (i).

From the above reasonings, we believe that sublinearly growing queues are as good as the bounded queues from a practical perspective. The following analysis is based on the concept of sublinearly growing queue lengths.

Definition 2: An actual queue length $Q_k(t)$ *grows sublinearly* if for any $\epsilon > 0$ and $\delta > 0$, there exists t_0 such that

$$\mathsf{Prob}(|Q_k(t)| > \epsilon t) < \delta,\ \forall t > t_0. \tag{6}$$

An SPN is *sublinearly stable* if all the queues grow sublinearly.

For any (0,1) random SPN, we have

Proposition 3: A rate vector $\mathbf{R}$ can be sublinearly stabilized only if there exist $\mathbf{s}_c \in \Lambda$ for all $c \in \mathsf{CQ}$ such that

$$\mathcal{A} \cdot \mathbf{R} + \sum_{c \in \mathsf{CQ}} f_c \cdot \overline{\mathcal{B}(c)} \cdot \mathbf{s}_c = 0. \quad (7)$$

Proposition 4: Consider any rate vector $\mathbf{R}$ such that there exist $\mathbf{s}_c \in \Lambda^\circ$ for all $c \in \mathsf{CQ}$ satisfying (7) holds. Then there exists an $\epsilon > 0$ such that the proposed scheme in Section IV-B can sublinearly stabilize the SPN with arrival rate $\mathbf{R}$.

Proposition 3 can be derived by simple flow conservation arguments. The main challenge of proving Proposition 4 is that the scheduling decision is based on the virtual queue lengths $q_k(t)$, which is updated in a way that is highly decoupled from the update rule of the actual queue lengths $Q_k(t)$. However, we need to prove that the scheduling based on $q_k(t)$ can sublinearly stabilize $Q_k(t)$. Several key ingredients of our analysis are provided as follows.

For discussion only, we temporarily ignore the ϵ-augmentation by setting $\epsilon = 0$. We then let each queue k keep another real-valued counter $q_k^{\text{inter}}(t)$, termed the *intermediate virtual queue length*. Initially, q_k^{inter} is set to 0. Recall that we make our scheduling decision based on $q_k(t)$ but we update $q_k^{\text{inter}}(t)$ by

$$\mathbf{q}^{\text{inter}}(t+1) = \mathbf{q}^{\text{inter}}(t) - \mathcal{A} \cdot \mathbf{a}(t) - \mathcal{B}(\mathsf{cq}(t)) \cdot \mathbf{x}^*(t). \quad (8)$$

That is, $\mathbf{q}^{\text{inter}}(t)$ is updated based on the actual realization of the service matrix. (Recall that each entry of $\mathcal{B}(\mathsf{cq}(t))$ is either 0 or 1.) $\mathbf{q}^{\text{inter}}(t)$ can strictly negative when updated via (8). We can then prove that the three quantities $q_k(t)$, $q_k^{\text{inter}}(t) - q_k(t)$, and $Q_k(t) - q_k^{\text{inter}}(t)$ all grow sublinearly. This in turns prove the sublinear growth of $Q_k(t)$.

V. The Combined Solution

We are now ready to combine the discussions in Sections III and IV. As discussed in Section III, the 7 operations form a vr-network as described in Fig. 5. Specifically, there are $K = 5$ queues, $M = 2$ IAs, and $N = 7$ SAs. The input matrix $\mathcal{A}$ contains 2 (negative) ones, since the packets arrive at either $Q_\emptyset^1$ or $Q_\emptyset^2$. Given the channel quality $\mathsf{cq}(t) = c$, the average service matrix $\overline{\mathcal{B}(c)}$ can be derived from Table II. We can then compute $\overline{\mathcal{B}(c)}^\epsilon$. Since there are 7 coding operations (SAs), each vector in $\mathfrak{X}$ is a 7-dimensional binary vector. Since we are allowed to choose any one of the 7 operations or choose to transmit nothing, 7 of the 8 vectors are the dirac delta vectors and the rest is an all-zero vector. We can now use (2), (4), and (5) to make the scheduling decision.

Proposition 5: When $\epsilon \to 0$, the sublinear stability region of the proposed INC-plus-SPN-scheduling scheme matches the block-code capacity of time-varying channels.

Sketch of the proof: Proposition 4 allows us to explicitly write the sublinear stability region by the linear equalities specified in by $\mathcal{A}$ and $\overline{\mathcal{B}(c)}$. We then compare the sublinear stability region polytope with the block-code capacity region polytope specified in Proposition 1. We can show that both polytopes are identical, which completes the proof.

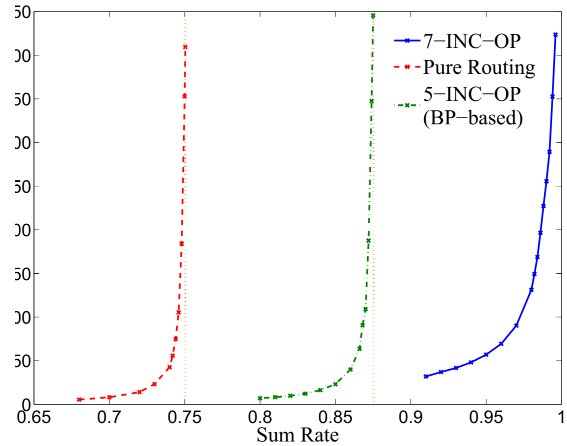


Fig. 6. The backlog of three different schemes for a time-varying channel with $\mathsf{cq}(t)$ uniformly distributed on $\{1, 2\}$, and the packet delivery probability being $\vec{p}^{(1)} = (0, 0.5, 0.5, 0)$ and $\vec{p}^{(2)} = (0, 0, 0, 1)$. The solid line represents the proposed 7-operation INC solution, the dotted-dash line represents the existing 5-operation solution [9], and the dash line represents the back-pressure-based pure-routing solution

Remark: When conducting simulations, we notice that we can make the following revisions to further reduce the actual queue lengths $Q_k(t)$ by $\approx 50\%$ even though we do not have any rigorous proofs for the performance of the revised scheme. The revision includes (i) We can simply set $\epsilon = 0$ in practice since the use of ϵ is mainly for circumventing some mathematical difficulties. (ii) When making the scheduling decision by (2), we can compute $\mathbf{d}(t)$ by

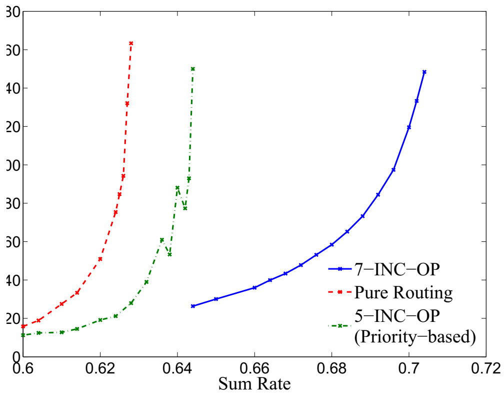
$$\mathbf{d}(t) = \left(\overline{\mathcal{B}(\mathsf{cq}(t))} \right)^T \mathbf{q}^{\text{inter}}(t). \quad (9)$$

where $\mathbf{q}^{\text{inter}}(t)$ is the intermediate virtual queue length defined in Section IV-C. This allows the scheme to directly control $q_k^{\text{inter}}(t)$, which is closely related to the actual queue length $Q_k(t)$.

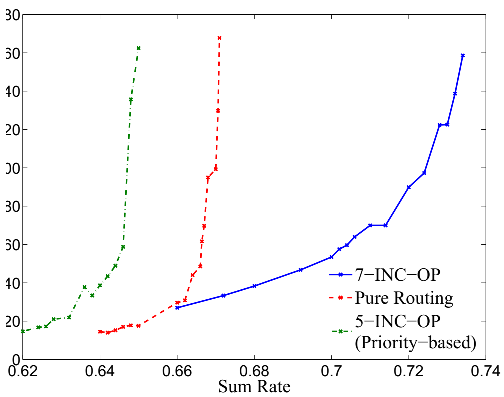
VI. Simulation Results

In this section, we simulate the 7-INC-operation solution proposed in Section III with the DMW-based scheduling algorithm in Section IV-B and V, and compare the stability results with the existing INC solutions and the (back-pressure) pure-routing solution to evaluate the performance improvement.

In Fig. 6, we simulate a simple time-varying channel situation first described in Section II-B1. Specifically, the channel quality $\mathsf{cq}(t)$ is i.i.d. distributed and for any t, $\mathsf{cq}(t)$ is uniformly distributed on $\{1, 2\}$. When $\mathsf{cq}(t) = 1$, the success probabilities are $\vec{p}^{(1)} = (0, 0.5, 0.5, 0)$ and and when $\mathsf{cq}(t) = 2$, the success probabilities are $\vec{p}^{(2)} = (0, 0, 0, 1)$, respectively. By the results in [12], the theoretical rate bound in this setting is 1 packet/slot for the optimal 7-operation INC scheme and 0.875 packet/slot for the suboptimal 5-operation scheme. One can also show that when using routing-based solutions (no network coding), the stability region is bounded by 0.75 packet/slot. The simulation results confirm



(a) i.i.d. channel quality.



(b) periodic channel quality.

Fig. 7. The backlog comparison for 4 different channel qualitiess. The solid line represents the proposed 7-operation INC solution, the dotted-dash line represents the priority-based 5-operation scheme [4], and the dash line represents the pure-routing solution.

our analysis. The proposed 7-operation dynamic INC design provides 14.7% throughput improvement over the 5-operation INC, and 33.3% over the pure-routing solution.

Next we simulate the scenario of 4 different channel qualities: $\mathsf{CQ} = \{1, 2, 3, 4\}$. The varying channel qualities could model the situations like the different packet transmission rates and loss rates due to adaptive coding and modulation; or the time-varying interference caused by the primary traffic in a cognitive radio environment. As will be seen, the proposed solution robustly and optimally adapt the time-varying channel quality and consistently outperforms all the existing solutions. In Fig. 7. We assume four possible channel qualities with the probability distributions are $\vec{p}^{(1)} = (0.14, 0.06, 0.56, 0.24)$, $\vec{p}^{(2)} = (0.14, 0.56, 0.06, 0.24)$, $\vec{p}^{(3)} = (0.04, 0.16, 0.16, 0.64)$, and $\vec{p}^{(4)} = (0.49, 0.21, 0.21, 0.09)$. In Figure 7(a), the channel quality $\mathsf{cq}(t)$ is i.i.d. distributed with the expected long-term frequency (f_1, f_2, f_3, f_4) being $(0.15, 0.15, 0.35, 0.35)$. In Figure 7(b) we consider the same $\vec{p}^{(1)}$ to $\vec{p}^{(4)}$ but choose $\mathsf{cq}(t)$ to be periodic with period 12 and the first period being $1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4$. As depicted by Figure 7(a) and 7(b), the pure-routing solution outperforms the 5-operation scheme for the periodic $\mathsf{cq}(t)$ while the order is reversed for i.i.d. $\mathsf{cq}(t)$. The proposed 7-operation scheme consistently outperforms all the existing solutions and achieves the optimal throughput.

VII. Conclusion

We have proposed a new 7-operation INC scheme together with the corresponding scheduling algorithm to achieve the optimal throughput of downlink 2-flow with time varying channels. Based on binary XOR operations, the proposed solution admits ultra-low encoding/decoding complexity. A byproduct of this paper is a throughput-optimal scheduling solution for SPNs with random departure, which further broadens the applications of SPNs to other real-world applications.

References

[1] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb 2003.

[2] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *Information Theory, IEEE Transactions on*, vol. 55, no. 2, pp. 797–815, 2009.

[3] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network," in *Proc. ACM Special Interest Group on Data Commun. (SIGCOMM)*, 2006.

[4] Y. Sagduyu, L. Georgiadis, L. Tassiulas, and A. Ephremides, "Capacity and stable throughput regions for the broadcast erasure channel with feedback: An unusual union," *Information Theory, IEEE Transactions on*, vol. 59, no. 5, pp. 2841–2862, 2013.

[5] C.-C. Wang, "On the capacity of 1-to-K broadcast packet erasure channels with channel output feedback," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 931–956, Feb 2012.

[6] ——, "On the capacity of wireless 1-hop intersession network coding — a broadcast packet erasure channel approach," *IEEE Trans. on Information Theory*, vol. 58, no. 2, pp. 957–988, Feb 2012.

[7] A. Eryilmaz, D. Lun, and B. Swapna, "Control of multi-hop communication networks for inter-session network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1092–1110, Feb. 2011.

[8] "Scheduling with pairwise xoring of packets under statistical overhearing information and feedback," *Queueing Systems*, vol. 72, no. 3-4, 2012.

[9] S. A. Athanasiadou, M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Stable and capacity achieving xor–based policies for the broadcast erasure channel with feedback," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. IEEE, 2013.

[10] L. Georgiadis and L. Tassiulas, "Broadcast erasure channel with feedback — capacity and algorithms," in *Proc. 5th Workshop on Network Coding, Theory, & Applications (NetCod)*, Lausanne, Switzerland, June 2009, pp. 54–61.

[11] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.

[12] C.-C. Wang and D. J. Love, "Linear network coding capacity region of 2-receiver mimo broadcast packet erasure channels with feedback," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 2062–2066.

[13] L. Jiang and J. Walrand, "Stable and utility-maximizing scheduling for stochastic processing networks," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE, 2009, pp. 1111–1119.

[14] L. Huang and M. J. Neely, "Utility optimal scheduling in processing networks," *Performance Evaluation*, vol. 68, no. 11, pp. 1002–1021, 2011.

[15] L. Amini, N. Jain, A. Sehgal, J. Silber, and O. Verscheure, "Adaptive control of extreme-scale stream processing systems," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*. IEEE, 2006, pp. 71–71.

[16] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, 2008, pp. 29–42.