Context-Aware Interflow Network Coding and Scheduling in Wireless Networks

Tuan Tran, *Member, IEEE*, and Thinh Nguyen, *Member, IEEE*

***Abstract*—Recent approaches using network coding (NC) to mix data from different flows show significant throughput improvement in wireless networks. However, in this paper, we argue that exhaustively mixing packets from different flows may decrease network quality of service (QoS), particularly in the presence of flows with different service classes. We therefore propose a *context-aware interflow network coding and scheduling* (CARE) framework, which adaptively encodes data across the traffic to maximize the network QoS. First, we develop a perception-oriented QoS (PQoS) to measure the user satisfaction of different types of services. Next, based on the characteristics of the traffic, we optimally combine data across the flows and schedule the encoded packets in each time frame to maximize the PQoS at the receivers. Solving CARE is NP-hard; thus, we devise a computationally efficient approximation algorithm based on the Markov chain Monte Carlo method to approximate the optimal solution. We prove that the proposed approximation algorithm is guaranteed to converge to the optimal solution. The analytical and simulation results show that, under certain channel conditions, the proposed CARE-based schemes not only improve the network QoS but achieve high throughput across all receivers as well. Additionally, the results show that the approximation algorithm is efficient and robust to the number of data flows. In some transmission conditions, our CARE-based schemes can improve the network QoS up to 50% compared with the existing randomized NC techniques.**

***Index Terms*—Multiuser multiservice scheduling, quality of service (QoS), random network coding (RNC), wireless networks.**

I. Introduction

We consider the problem of downlink transmission in wireless networks, whereby multiple data flows share a single wireless channel. Traditionally, data transmission is performed via the *store-and-forward* routing protocols in which an intermediate node stores incoming data and forwards them to the neighboring nodes toward the destinations without altering contents of the data. Differently, in the new *network coding* (NC) approach [1], an intermediate node is allowed to combine the incoming data packets before sending them out to the receivers. It has been shown that NC-based approaches significantly improve network performance, such as throughput [2]–[6], transmission delay [3], [7], and energy [8], [9]. For instance, Nguyen *et al.* [2], [6] showed that XOR-based NC schemes used in conjunction with a scheduler at a WiFi access point (AP) can significantly improve the network throughput of a broadcast session. The authors showed that, under some transmission conditions, bandwidth efficiency could be double compared with the traditional Auto Repeat reQuest (ARQ) approaches [10], [11]. Additionally, Eryilmaz *et al.* [3] have shown that, in unreliable wireless networks, transmission delay decreases substantially by using NC. Furthermore, Tran *et al.* [12], [13] showed that significant bandwidth gain can also be achieved by employing NC in conjunction with channel coding techniques across different unicast sessions. In a different avenue, Wu *et al.* [8] showed that NC can also be used to minimize the transmission energy in mobile ad-hoc networks as well.

In this paper, we focus on using NC-based approaches to improve the quality of service (QoS) of wireless networks that consist of multiple unicast flows. Generally speaking, providing high QoS for multiuser wireless networks is challenging. First, wireless links often suffer due to severe fading and interference. Additionally, channel conditions that usually change over time significantly affect the QoS at the receivers. Furthermore, the heterogeneity of channel conditions makes the scheduling problem more challenging, as receivers usually experience different data losses. Retransmitting lost data to a receiver in a bad channel condition may decrease the network bandwidth efficiency because data could be duplicated at other receivers. On the other hand, only serving receivers in good channel conditions could leave many other receivers in worse channel conditions with unacceptable QoS. The problem usually becomes combinatorially hard in nature.

In fact, there exists literature on using NC to improve network QoS. Notably, Seferoglu *et al.* [14], [15] proposed video-aware opportunistic XOR-based network encoding and scheduling schemes for video streaming. The proposed schemes take into account both video distortion and deadlines of the packets for optimal data encoding and scheduling. The simulation results showed significant video quality improvement [i.e., peak-signal-to-noise ratio (PSNR)] compared with the approaches without video-aware encoding. Those works, however, considered only the case where all data flows are multimedia streams (i.e., video). As a result, they may not work well when applied directly to scenarios where traffic includes both delay-sensitive (e.g., video streaming) and elastic applications (e.g., web browsing). In such a setting, the performance metric of

Manuscript received October 31, 2014; revised April 7, 2015, July 20, 2015, and November 17, 2015; accepted January 1, 2016. This work was supported in part by the National Science Foundation under Grant CNS-0845476 and Grant CNS-1547450 and in part by the Faculty Scholarship Grant of Sullivan University. This paper was presented in part at the 19th IEEE International Conference on Computer Communications and Networks (ICCCN), 2010. The review of this paper was coordinated by Prof. C. Assi.

T. Tran is with the College of Information and Computer Technology, Sullivan University, Louisville, KY 40205 USA (e-mail: ttran@sullivan.edu).

T. Nguyen is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA (e-mail: thinhq@eecs.oregonstate.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2016.2520361

delay-sensitive applications (e.g., PSNR for video streaming) might not be a good metric to measure the QoS of the delay-tolerant applications. Generally, the heterogeneity of the traffic content makes the encoding and scheduling problem more challenging because the transmitter needs to consider not only the transmission deadline but the *characteristics of the traffic* as well.

We consider the problem of multiuser downlink transmission of heterogeneous traffic in lossy wireless networks. A typical approach for such a system is to overprovision the QoS requirements for various flows. Such a system will work well as long as the aggregate demand from all the flows does not exceed the network capacity [16]. However, in an overloaded transmission scenario, the system performance is likely to deteriorate rapidly due to a traffic surge [17]. Therefore, we propose a new NC-based framework called Context-awARE Interflow Network Coding and Scheduling (CARE) which utilizes the user perception-oriented QoS model (PQoS) [18] for optimal interflow data encoding and scheduling. Generally speaking, CARE optimally encodes and schedules data transmission based on both channel conditions and characteristics of the traffic to maximize the network QoS from the user's perspective. Our results show that CARE significantly improves the network QoS and throughput, in comparison with the state-of-the-art NC-based approaches. The performance improvement basically comes from the optimal trade-off between the number of useful data packets transmitted by the deadline and the bandwidth allocation to different flows via its PQoS objective function. To the best of our knowledge, this is one of a few papers that consider PQoS as the objective function in formulating the multiuser downlink scheduling using NC in wireless networks.

Extending from our preliminary results in [19], the main contributions of this paper are summarized as follows.

- We first show that the approaches optimized for throughput might decrease the QoS at some receivers in the presence of traffic with different service classes. We then devise PQoS as the network metric to quantitatively measure the performance of different transmission strategies. The CARE framework based on PQoS consolidates not only the traffic priorities but also the characteristics of the flows in its encoding and scheduling operation. Analyses on QoS performance for different transmission strategies are provided in detail.
- We formulate CARE as a combinatoric optimization problem with constraints. We further show that CARE can be reduced to a set of weighted stochastic knapsack problems and, therefore, is NP-hard [20]. We then exploit the traffic characteristics to devise an efficient approximation algorithm based on the Markov chain Monte Carlo (MCMC) method for finding a near-optimal solution. Our results show that, under certain channel conditions and QoS requirements, our proposed CARE-based schemes lead to significant network performance improvement for both delay-sensitive and delay-tolerant data flows. We further provide analytical results on the asymptotic behavior and upper bound on the convergence time of the approximation algorithm based on the canonical path technique [21]. We also describe context-aware partial interflow NC (PCARE), an extension of CARE, which allows for finer grained bandwidth allocation.
- We provided theoretical analysis and intensive simulations to elaborate the system performance improvement of our proposed schemes. Our results reveal that optimizing the PQoS is equivalent to optimizing the network effective throughput constrained on the fairness condition among the users. The results also show that the approximation algorithm is efficient and robust to the number of data flows and quickly converges to the optimal solution.

The remainder of this paper is organized as follows. We first discuss some background and related work in Section II. In Section III, we describe the system model, the issues of the existing approaches, and performance metric. In Section IV, we analyze the system performance of different transmission strategies, CARE formulation, and its hardness. In Section V, we develop an approximation algorithm for CARE. Simulations and discussions are provided in Section VI. Finally, we conclude this paper in Section VII.

II. Background and Related Work

Random Network Coding: The notion of NC, i.e., mixing of data at intermediate nodes to increase the overall multicast throughput of a network, was first proposed in the seminal paper by Ahlswede *et al.* [1]. Its key idea is to prove the existence of some network codes (method of mixing data at intermediate nodes) that achieve multicast capacity. This spurted a number of works on construction of practical network codes, including algebraic, algorithmic, and randomized approaches [22]–[25]. In particular, the work in [22] proposed a class of linear network codes for multicast transmission. The author proved that linear coding suffices to achieve the maximum throughput, which is the max-flow min-cut from the source to each receiving node. Inspired by this work, Koetter and Medard [23] proposed a theoretical framework based on algebraic tools for deriving the conditions to achieve capacity in networks using linear codes. The proposed framework shows interesting connections between certain systems of polynomial equations and the solutions to network routing problems. Notably, the work in [4] (and its extended version in [26]) proposed a random NC (RNC) framework for efficient network code construction in a distributed manner. In particular, it shows that intermediate nodes in a network do not need to cooperate with each other to generate coded packets. Instead, each node independently generates its coded packets by combining its incoming data with the coding coefficients randomly selected from a large finite field. The authors proved that, with probability approaching 1, the encoded packets are independent. Based on this result, Chou *et al.* [25] proposed a practical solution to implement NC for an arbitrary network topology. In particular, it has been shown that, by inserting the coding coefficients into the coded packets' headers, the sinks can reconstruct the original data efficiently by solving a system of linear equations constructed by the encoded data.

Time slot	1		2		3		4
Scheme	UNI	RNC	UNI	RNC	UNI	RNC	UNI
Packet	*a*	c_1	*b*	c_2	*a*	c_3	b
D_1	x	x	-	o	o	o	-
D_2	-	o	x	x	-	o	o

Fig. 1. Example of UNI and RNC transmission schemes for two receiver scenarios. For the RNC scheme, coded packets $c_i = \alpha_i a + \beta_i b$. Packet receipts are denoted by "$\times$," "o," and "-" for lost, successful, and "received but useless" packets, respectively.

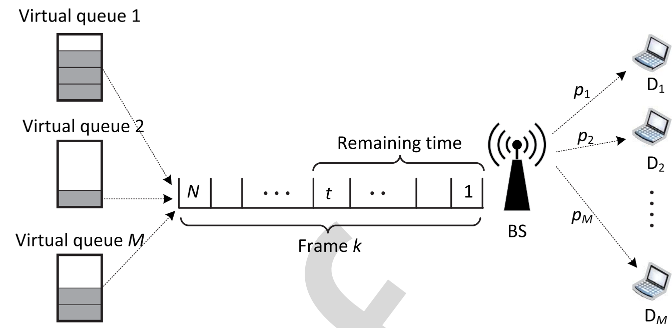


Fig. 2. System model. At the beginning of frame k, new arrived packets will be scheduled for transmission in the next N time slots (period of a frame).

RNC for Wireless Networks: It has been shown that applying NC in wireless ad-hoc networks can significantly improve the network bandwidth efficiency [27], transmission delay [28], [29], or transmission energy [7]. The key idea of these works is to exploit the nature of a broadcast signal, which can be intercepted by many neighboring nodes. Data of different flows are then combined (i.e., performing RNC) together to generate coded data packets before sending them to other nodes in the vicinity. Recently, Douik *et al.* [30], [31] have proposed techniques to reduce the decoding delay for different feedback constraints. These works, however, only focus on broadcasting transmission and do not consider the flows with heterogeneous content and services. A detailed list of wireless applications beneficial from using NC in these settings can be found in [29].

Our network model is closest to the model used in [2] and [6], whereby the authors model a broadcast session in a last-mile network. Differently, we consider multiple unicast flows sharing a bandwidth-constrained channel. Our transmission model can be applied to many practical transmission scenarios in WLAN, WiMAX, and cellular networks. Before describing our system model, we first illustrate the benefit of using NC in such a setting.

Example 1: An AP wishes to send two packets a and b to two receivers D_1 and D_2, respectively. In ARQ unicast protocol (UNI), the AP uses the first and second time slots to send a and b, respectively. As illustrated in Fig. 1, packet a is lost at D_1 while successfully received at D_2. However, D_2 discards a because it wants packet b instead. Similarly, in the second time slot, packet b is successfully received by D_1 (but it is discarded because D_1 wants a instead) while lost at D_2. Assuming that, in the next two time slots, the transmission links are in good conditions, then the AP can successfully retransmit the lost data packets to their intended receivers. As a result, it needs four time slots to deliver two packets a and b to D_1 and D_2, respectively.

Next, we consider a transmission scheme using RNC, as proposed in [32]. In this approach, the AP generates coded packets by linearly combining a and b with random coefficients and sending them out to the receivers. For example, coded packets c_i are generated as $c_i = \alpha_i a + \beta_i b$, $i = \{1, 2, 3, \ldots\}$, where α_i and β_i are coefficients drawn randomly from a large finite field F_q (q is the field size). The AP then broadcasts two coded packets c_1 and c_2 in the first and second time slots, respectively, as shown in Fig. 1. As a result, D_1 and D_2 will successfully receive c_2 and c_1, respectively. In the third time slot, the AP broadcasts another coded packet c_3, which is successfully received by both receivers. After three transmissions, each of the receivers now has two coded packets, which can be used to recover their desired data by solving a system of linear equations using the encoded packets. We note that the coefficients are included in the packets' headers, enabling the receivers to solve the linear equation system. It requires additional transmission overhead; however, with a sufficiently large packet size, this overhead is negligible [25]. Thus, the RNC approach needs only three transmissions to deliver two packets to the receivers, saving 25% transmission bandwidth compared with the UNI scheme.

III. System Model, Issues, and Performance Metric

A. System Model

We consider the problem of time-slotted downlink transmission of M data flows to M receivers (users) in lossy wireless networks, as illustrated in Fig. 2. We assume that the transmitter uses M "virtual queues" to store randomly arrived packets of different flows before sending them out to the corresponding receivers. Transmissions are scheduled by frames (periods), each consisting of N time slots. Generally, the value of N can be determined by the hard deadline of buffered data or by the number of backlogged data packets in the buffer [15], [33]–[35]. When a new flow arrives at the transmitter in the current transmission frame, it will be scheduled for transmission in the next frame. Without loss of generality, we assume that there are $k_i, i = 1, 2, \ldots, M$ data packets being delivered to receiver D_i in each transmission frame. In this paper, we focus on finding an optimal flow partition scheme for encoding and scheduling data across different flows, given a set of data packets for each transmission frame.

We assume that the scheduled packets of delay-sensitive applications, which cannot be delivered to the corresponding receivers by the deadline, will be discarded from the system. The system QoS is computed based only on the number of data packets that have been received successfully within that period. On the other hand, for delay-tolerant reliable applications such as file transfer, the lost packets will be retransmitted until they are successfully delivered (possibly via multiple transmission frames). Detail of the mathematical formula used to compute the system QoS is defined in Section III-C3. In addition, we assume that the transmission links between the transmitter and the receivers are heterogeneous and independent identically distributed binary erasure channels with erasure probability p_i. A flow i implements a packet-level forward error-correcting (FEC) code (n_i, k_i) (e.g., [36]) to cope with data errors. Using such an FEC code, receiving k_i out of n_i transmitted packets is sufficient for receiver D_i to recover the original information. The code rates k_i/n_i are prespecified based on the network conditions or/and priority of the users' subscription. Finding the optimal coding rates is beyond the scope of this paper. Additionally, we assume that the transmitter has enough memory to store data for at least one transmission frame $N = \sum_{i=1}^{M} n_i$.

Furthermore, in our model, each receiver maintains a *decoding buffer* for storing coded packets received within a frame. By the end of each data frame, a receiver will decode the received packets and push them to the upper Open Systems Interconnection (OSI) layer above for further processing. It is important to note that the use of a decoding buffer at receivers is a standard assumption, which has been adopted for several years in NC-based techniques and literature (e.g., [5], [6], [15], [26], [30], [37], [38], and references therein).

B. What Could Go Wrong With Interflow Network Coding?

As observed in *Example 1*, mixing packets from different flows is clearly beneficial. However, *should one advocate mixing at every opportunity*? The answer should be "No." In particular, Wu *et al.* were the first to consider this mixing problem in a different context [39]. Intuitively, exhaustively mixing the data of several flows may decrease the chance that a receiver can recover its information. To illustrate this point, we show a simple counterexample as follows.

Counterexample: We consider the same setup, as shown in Fig. 1. In addition, we assume that packet losses at D_1 and D_2 are independent and follow the Bernoulli trial with $p_1 = 1/3$ and $p_2 = 2/3$, respectively. To transmit data reliably, it makes sense for the transmitter to employ stronger protection for the data intended to D_2. Thus, suppose that two packet-level FEC codes $(n_1, k_1) = (3, 2)$ and $(n_2, k_2) = (3, 1)$ are used for the flows to D_1 and D_2, respectively.[1] We recall that, by using a code (n, k), the transmitter uses n time slots to transmit k information packets $(n \geq k)$, whereby receiving any k packets out of the n transmitted packets is sufficient to recover the k original packets. Suppose that the transmitter has $n_1 + n_2 = 6$ time slots for delivering three packets, i.e., two for D_1 and one for D_2. In a non-mixing technique, i.e., packets from different flows are sent separately, the probability that all the receivers recover their desired data is computed as

$$P_{\text{UNI}} = \prod_{i=1}^{2} \sum_{j=0}^{r_i} \binom{n_i}{j} p_i^j (1 - p_i)^{n_i - j} \tag{1}$$

where $r_i = n_i - k_i$ for $i = \{1, 2\}$. Substituting values of p_i, n_i, and k_i into (1), we have $P_{\text{UNI}} = 0.5213$, which is about 1/2 packet per time slot.

On the other hand, in an RNC-based technique, all packets are mixed to produce coded packets. In such a transmission strategy, each receiver needs to receive at least three coded packets correctly, to recover its desired information [41]. The transmitter has six time slots for delivering the *coded* packets to both receivers. The probability that both receivers recover their desired data is given by

$$P_{\text{RNC}} = \prod_{i=1}^{2} \sum_{j=0}^{r} \binom{N}{j} p_i^j (1 - p_i)^{N - j} \tag{2}$$

where $r = \sum_{i=1}^{2} (n_i - k_i) = 3$ and $N = \sum_{i=1}^{2} n_i = 6$. Substituting the values of p_i for $i = \{1, 2\}$ into (2), we obtain $P_{\text{RNC}} = 0.2876$. This is approximately equivalent to a throughput of 1/5 packet per time slot, which is about 2.5 times less than that of the non-NC technique earlier. Obviously, applying NC in this case decreases the network throughput.

C. Perception-Oriented QoS

We next describe how the PQoS metric is constructed to measure the performance of different transmission strategies. Roughly speaking, PQoS function is used to estimate the service satisfaction at each user. Thus, PQoS depends not only on the number of received packets within a time period but also on the type of service (ToS). For the sake of exposition, we categorize the network traffic into two types: delay-sensitive traffic (e.g., video streaming, audio streaming, etc.) and elastic traffic (e.g., file transfer, e-mail, etc.).[2] We note, however, that one can easily extend the framework to the cases of more than two types of traffic, albeit a more sophisticated model.

1) PQoS for Delay-Sensitive Traffic: In our model, delay-sensitive traffic (e.g., video streaming) is encoded into multiple layers, e.g., a base layer and enhancement layers [42], [43]. In such a model, the base layer must be presented to present other enhancement layers. Thus, to maintain a minimal QoS, a receiver needs to receive at least N_0 packets of the base layer per transmission frame. When the number of received packets is fewer than N_0, the QoS at the receiver decreases significantly. The reason is that, in such an encoding scheme, the QoS at the receiver is mainly contributed by the base layer. On the other hand, when more packets of the enhancement layers are received, the QoS at the receiver only increases slightly due to only additional details of the information added into the base frame. We extend the satisfaction function proposed in [18] to model the PQoS for delay-sensitive applications. Mathematically, we can represent it in (3), shown at the bottom of the next page, where S_i denotes the number of packets received successfully, and N_0 denotes the minimum number of packets to maintain a satisfaction factor of $\gamma_0 \in [0, 1]$. In this formulation, the first case indicates that, when the number of received packets is fewer than the threshold $\gamma_0 N_0$, the value of PQoS is equal to zero. The intuition can be reasoned in the context of layered video transmission (e.g., MPEG-2) where the base layer is unrecoverable. Thus, no information is displayed,

[1] We note that the packet-level FEC codes here can be viewed as the raptor codes [40] with n_i and k_i, respectively, being the output and original symbols.

[2] We use the terms "elastic" and "delay-tolerant" interchangeably.

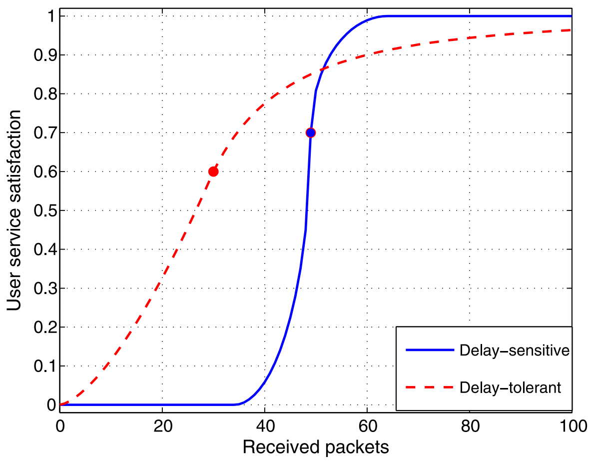


Fig. 3. Satisfaction functions of different applications versus the number of received packets.

resulting in a zero PQoS. The second case accounts for the scenario when the number of received packets is greater than the minimum threshold $\gamma_0 N_0$ but less than N_0. In this case, the PQoS steeply decreases due to only a part of the base layer recovered. The third case indicates that the value of PQoS increases significantly when the number of received packets is greater than N_0. In this case, more detail of the video frame is added to achieve high-definition display. Finally, the last case accounts for a scenario wherein the missing data add negligible QoS to the data flow, resulting in a high value of PQoS close to one. An example of the PQoS function for delay-sensitive application, with respect to the number of received packets, is illustrated by the blue solid curve in Fig. 3.

2) PQoS for Delay-Tolerant Traffic: We use different functions to model elastic traffic, as illustrated by the red dashed curve in Fig. 3. For such a flow, the parameter N_0, i.e., the minimum data received per frame, is viewed as the minimum bandwidth allocated to that flow. When the number of received packets is fewer or greater than N_0, respectively, the satisfaction function decreases or increases slightly. We note that, in delay-tolerant traffic, if a transmitted packet is lost during transmission, a negative acknowledgement (NACK) will be sent from the receiver to the transmitter for retransmission (possibly in other frames). Thus, for reliable data flows (e.g., data file downloading), every byte of the information will be reliably delivered to the receiver. Mathematically, the PQoS for elastic traffic is given as

$$\gamma_i = \mathcal{F}(S_i) = \begin{cases} \gamma_0 \cdot \left(\frac{S_i}{N_0}\right)^2, & S_i < N_0 \\ 1 - (1-\gamma_0)\left(\frac{N_0}{S_i}\right)^2, & S_i \geq N_0. \end{cases} \tag{4}$$

In the first case, when the number of received packets is fewer than N_0, the value of PQoS decreases slightly by a factor of the ratio of S_i and N_0. It is worth noting that this ratio is less than one; thus, its square would result in a smaller value. On the other hand, when $S_i \geq N_0$, PQoS increases slightly. The intuition of the proposed PQoS functions is to reflect the fact that a slightly longer or shorter delay in receiving a delay-tolerant data packet, e.g., e-mail, does not affect much to the user's perception of the QoS.

3) Performance Metric: We use the average network PQoS as our metric to compare the performance of different transmission strategies. The average network PQoS is computed by averaging the expected PQoS across all the users for each transmission frame over infinite system runtime. Mathematically, we have that

$$\gamma = \lim_{\tau \to \infty} \frac{1}{\tau M} \sum_{\tau} \sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \tag{5}$$

where τ is the operation time of the network divided into several transmission frames, M is the number of data flows, c_i is the flow priority, and $\mathbb{E}[\gamma_i]$ denotes the expected PQoS of user i in one transmission frame. The $\lim_{\tau\to\infty}$ indicates that the expectation of the system performance is computed over a long time interval. Given the satisfaction functions, we call a transmission technique the best, if it has the largest expected PQoS across all users.

IV. Performance Analysis

In this paper, we analyze different transmission strategies, depending on how the transmitter exploits the characteristics of the traffic. In particular, we analyze the performance of the traditional UNI, systematic RNC (SRNC), and CARE.

A. Unicast

In this technique, data packets of different information flows are transmitted separately in a round-robin fashion [44]. In each period (frame), the transmitter allocates n_i time slots to transmit k_i data packets to receiver D_i [i.e., packet-level FEC code (n_i, k_i)]. If there is a packet loss, the transmitter uses the $n_i - k_i$ redundant time slots to retransmit the lost packets. The transmitter switches to transmit data packets for other users, when it receives an acknowledgment (ACK) message from D_i indicating that all data have been received successfully, or all time slots allocated for it have been used. Considering flow i destined to receiver D_i and letting p_i denote the packet erasure

$$\gamma_i = \mathcal{F}(S_i) = \begin{cases} 0, & S_i < \gamma_0 N_0 \\ \gamma_0 \left(1 - \frac{\sqrt{(N_0-S_i)(N_0+S_i-2\gamma_0 N_0)}}{(1-\gamma_0)N_0}\right), & \gamma_0 N_0 \leq S_i < N_0 \\ \gamma_0 + (1-\gamma_0)\frac{\sqrt{(S_i-N_0)[N_0-S_i+2N_0(1-\gamma_0)]}}{(1-\gamma_0)N_0}, & N_0 \leq S_i < (2-\gamma_0)N_0 \\ 1, & S_i \geq (2-\gamma_0)N_0 \end{cases} \tag{3}$$

probability of the transmission link, the probability that receiver D_i recovers all its data can be written as

$$P_i^s = \sum_{j=k_i}^{n_i} \binom{n_i}{j} p_i^{n_i-j}(1-p_i)^j \tag{6}$$

where $\binom{n_i}{j}$ denotes the number of combinations of size j out of n_i elements. We further note that, by using UNI, i.e., data transmitted separately, receiver D_i could recover a fraction of the original data, if the number of received packets is fewer than k_i. Let the random variables X and Y, respectively, be the number of original packets and the total number of received packets. The probability that D_i obtains only m original packets and the total received packets is fewer than k_i is given as

$$\begin{aligned} P_i(m) &= P(X=m, Y<k_i) \\ &\overset{(a)}{=} \sum_{l=m}^{k_i-1} \Pr(Y=l|X=m)\Pr(X=m). \end{aligned} \tag{7}$$

In this case, Y is equal to X plus the number of coded packets received during the second stage transmission while the sum runs over all the possible values of Y. We note that step (a) consists of two parts: 1) the conditional probability of receiving $l = m, \ldots, k_i - 1$ data packets in total, given m original data packets received; 2) the probability that m of k_i original data packets are received. In the extreme case, i.e., $l = m$, it implies that none of the coded packets is received. On the other hand, the maximum number of coded packets is $k_i - 1 - m$, corresponding to the case of $k_i - 1$ data packets received in total. Furthermore, the probability that only m of k_i original data packets are received is written as

$$\Pr(X=m) = \binom{k_i}{m}(1-p_i)^m p_i^{k_i-m}. \tag{8}$$

Given that only m original data packets are received, the probability that only $l < k_i$ data packets are received over n_i transmissions is given by

$$\Pr(Y=l|X=m) = \binom{n_i-k_i}{l-m}(1-p_i)^{l-m} p_i^{n_i-k_i-(l-m)}. \tag{9}$$

Combining (8) and (9), we have that

$$\begin{aligned} &P(Y=l|X=m)P(X=m) \\ &= \binom{k_i}{m}(1-p_i)^m p_i^{k_i-m}\binom{n_i-k_i}{l-m} \\ &\quad \times (1-p_i)^{l-m} p_i^{n_i-k_i-(l-m)} \\ &= \binom{k_i}{m}\binom{n_i-k_i}{l-m}(1-p_i)^l p_i^{n_i-l}. \end{aligned} \tag{10}$$

Therefore, the probability that the m original data packets and the total $l < k_i$ data packets are received can be computed as

$$P_i(m) = \sum_{l=m}^{k_i-1} \binom{n_i-k_i}{l-m}\binom{k_i}{m} p_i^{n_i-l}(1-p_i)^l. \tag{11}$$

Let γ_i denote the PQoS value of receiver D_i; therefore, from (6) and (7), the expected PQoS across all the users can be written as

$$\begin{aligned} \gamma &= \frac{1}{M}\mathbb{E}\left[\sum_{i=1}^{M} c_i\gamma_i\right] = \frac{1}{M}\sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \\ &= \frac{1}{M}\sum_{i=1}^{M} c_i\left(\mathcal{F}(k_i)P_i^s + \sum_{m=0}^{k_i-1}\mathcal{F}(m)P_i(m)\right) \end{aligned} \tag{12}$$

where $c_i \in (0,1]$ denotes the weighted factor (priority) for the ith flow (this factor can be justified via the cost that the user D_i pays to the service provider); $\mathbb{E}[.]$ denotes the expected function; and $\mathcal{F}(\cdot)$ is defined in (3) and (4), depending on the type of the data flow.

B. Systematic Random Network Coding

Transmission in SRNC is classified into base and augmentation phases. In the base phase, all $\sum_{i=1}^{M} k_i$ original packets will be transmitted. The receivers cache all the received packets, including packets that are not intended to them. By keeping data packets intended to others, a receiver can use them to decode its desired data in the second phase. Next, in the augmentation phase, the transmitter combines the data packets of all flows to generate coded packets and broadcasts them to the receivers over $N - \sum_{i=1}^{M} k_i$ redundant time slots. The intuition of using systematic coding that transmits data in two phases is that the receivers that lose some original packets can receive more coded packets to decode their own data using the RNC method [41]. On the contrary, receivers that are unable to obtain a full set of data packets can still recover partial data from the original packets transmitted in the base phase. Such a transmission has been discussed in [45], and generally, it will result in higher performance compared to the RNC.

In SRNC, a receiver D_i can recover k_i desired packets, if it correctly receives either all k_i original packets or a full set of $K = \sum_{i=1}^{M} k_i$ packets (either original or coded packets). Let P_i^s denote the probability that D_i can recover all its data, and let the random variables U and V denote the original and total received packets at receiver D_i, respectively. We have that

$$\begin{aligned} P_i^s &= P(U=k_i, V<K) + P(U\le k_i, V\ge K) \\ &= (1-p_i)^{k_i}\left[\sum_{l=0}^{K-k_i-1}\binom{N-k_i}{l} p_i^{N-k_i-l}(1-p_i)^l\right] \\ &\quad + \sum_{j=0}^{k_i}\binom{k_i}{j} p_i^{k_i-j}(1-p_i)^j \\ &\quad \times \sum_{t=K-j}^{N-k_i}\binom{N-k_i}{t} p_i^{N-k_i-t}(1-p_i)^t. \end{aligned} \tag{13}$$

In (13), the first term accounts for the case when all the k_i original data packets are received successfully during the base phase transmission. The second term expresses the probability that j original packets are received during the base phase and

t coded packets are received during the augmentation phase, where $t = K - j, \ldots, N - k_i$. In this case, at least K data packets (both original and coded packets) are received by D_i; thus, it can recover its data by solving the system of linear equations formed by the received data packets.

We further note that D_i may receive only partial of the original data during the base phase. We denote $P_i(m)$ being the probability that D_i receives m original packets ($m < k_i$), and its total number of packets received correctly is less than K. Similar to (7), the probability of partial data recovery of D_i is written by

$$\begin{aligned} P_i(m) &= P(U = m, V < K) \\ &= \sum_{l=m}^{K-1} \binom{N-k_i}{l-m} \binom{k_i}{m} p_i^{N-l} (1-p_i)^l. \end{aligned} \quad (14)$$

In such cases, D_i cannot recover all its data, and only data received during the base phase contribute to the QoS of flow i. From (13) and (14), we express the expected PQoS across all receivers as follows:

$$\begin{aligned} \gamma &= \frac{1}{M} \mathbb{E}\left[\sum_{i=1}^{M} c_i \gamma_i\right] = \frac{1}{M} \sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \\ &= \frac{1}{M} \sum_{i=1}^{M} c_i \left(\mathcal{F}(k_i) P_i^s + \sum_{m=0}^{k_i - 1} \mathcal{F}(m) P_i(m) \right). \end{aligned} \quad (15)$$

C. Proposed Context-Aware Interflow Network Coding and Scheduling

1) Main Idea: Instead of combining all flows together, the transmitter now selectively chooses the flows to be mixed based on the channel conditions, ToS, and priorities of the flows. We assume that the M incoming data flows are partitioned into G groups; then, for each group, the transmitter uses SRNC to transmit data to the receivers within that group. The objective of the transmitter is to determine the optimal partition (i.e., which flows are combined together) to maximize the PQoS across all users. It is clear that mixing packets from all incoming flows could decrease the system performance due to *mismatch in ToS, priorities, and channel conditions*. On the other hand, mixing packets of flows with similar characteristics could increase the network performance. A precise mathematical formulation of CARE will be described as follows.

2) CARE Formulation: Let $\mathcal{G}$ denote a partition of the incoming information flows and $|\mathcal{G}|$ denote the number of groups in $\mathcal{G}$. Let M_i denote the number of data flows in group $i, i = 1, \ldots, |\mathcal{G}|$. Per each group, we use the SRNC technique described earlier to transmit the data. Consider the ith group, and let $N_i = \sum_{j=0}^{M_i} n_{ij}$ and $K_i = \sum_{j=0}^{M_i} k_{ij}$, respectively, denote the total number of available time slots and information packets being transmitted for group i. Here, (n_{ij}, k_{ij}) denotes the packet-level FEC of flow delivered to receiver j of group i. We note that flow j of group i, i.e., f_{ij}, is nothing but just some original data flow f_t, where the index has been relabeled. Thus, the FEC code (n_{ij}, k_{ij}) is also just a relabeled version of the original FEC code (n_t, k_t). Similarly, as computed in SRNC technique, the probability that receiver j of group i, i.e., D_{ij}, can recover its desired data is given as

$$\begin{aligned} P_{ij}^s = (1-p_{ij})^{k_{ij}} &\left[\sum_{l=0}^{K_i - k_{ij} - 1} \binom{N_i - k_{ij}}{l} p_{ij}^{N_i - k_{ij} - l} (1-p_{ij})^l \right] \\ &+ \sum_{s=0}^{k_{ij}} \binom{k_{ij}}{s} p_{ij}^{k_{ij}-s} (1-p_{ij})^s \\ &\times \sum_{t=K_i - s}^{N_i - k_{ij}} \binom{N_i - k_{ij}}{t} p_{ij}^{N_i - k_{ij} - t} (1-p_{ij})^t. \end{aligned} \quad (16)$$

In this equation, the first term accounts for the case where original packets are successfully received during the basis transmission phase of group i, whereas the second term expresses the probability that at least K_i data packets (including both original and coded packets) are received successfully after both phases of transmission.

Similarly, it is straightforward to calculate the probability that receiver D_{ij} recovers m out of k_{ij} original packets ($m < k_{ij}$). That is

$$P_{ij}(m) = \sum_{l=m}^{K_i - 1} \binom{N_i - k_{ij}}{l-m} \binom{k_{ij}}{m} p_{ij}^{N_i - l} (1-p_{ij})^l. \quad (17)$$

Let a random variable γ_{ij} denote the PQoS of receiver D_{ij}. Then, the expected PQoS over all users is given by

$$\gamma(\mathcal{G}) = \frac{1}{M} \mathbb{E}\left[\sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{M_i} c_{ij} \gamma_{ij} \right]. \quad (18)$$

Therefore, a partition scheme is optimal, if it maximizes the expected PQoS across all users. The CARE optimization problem can be formulated as

$$\begin{aligned} \text{CARE}: \quad & \max_{\mathcal{G} \in \Omega} \left\{ \frac{1}{M} \sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{M_i} c_{ij} \mathbb{E}[\gamma_{ij}] \right\} \\ \text{s.t.}: \quad & \sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{M_i} n_{ij} = N && (19) \\ & \sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{M_i} k_{ij} = K && (20) \\ & 0 \le c_{ij} \le 1 \text{ for } i = 1, 2, \ldots, |\mathcal{G}| \\ & j = 1, 2, \ldots, M_i && (21) \\ & \mathbb{E}[\gamma_{ij}] = \mathcal{F}(k_{ij}) P_{ij}^s + \sum_{m=0}^{k_{ij}-1} \mathcal{F}(m) P_{ij}(m) && (22) \end{aligned}$$

where Ω denotes the collection of all nonempty-subset partitions of M flows. The objective is to maximize the average expected PQoS across all the receivers. The first constraint represents the maximum number of time slots available for transmission in a data frame. The quantity $\sum_{j=1}^{M_i} n_{ij}$ gives the number of time slots allocated for group $i, i = 1, \ldots, |\mathcal{G}|$. The second constraint accounts for the total number of original data packets that need to be transmitted, i.e., $\sum_{j=1}^{M_i} k_{ij}$. Finally, the last two constraints, respectively, express the weight factor and the equation to compute the expected PQoS based on the type of applications and the number of received packets for receiver j in group i.

3) CARE Hardness: We next show that finding the optimal solution to the CARE problem is NP-hard. In particular, we will show 1) an existing algorithm that reduces an instance of CARE to an instance of the stochastic reward knapsack problem (SKP) [20] in polynomial time, and show that, 2) given the solution of CARE, the solution of the SKP can be found in polynomial time.

SKP Description: The SKP problem is described as follows. Given a set of n items, where item i has a fixed weight w_i and a random value v_i, the distribution of v_i could be unknown. The objective is to select a subset of items such that it maximizes the sum values while not exceeding the limit capacity of the knapsack.

Instance Reduction: We assume that there are M information flows, represented by their FEC codes $(n_i, k_i), i = 1, \ldots, M$, that we want to deliver to M receivers within N time slots. Without loss of generality, we assume that $\sum_{i=1}^{M} n_i > N$, i.e., only a subset of the M flows is selected. The expected PQoS of each subgroup of the M flows is considered as reward v_i of an item of the SKP problem. It is clear that $v_i \in V$ is a random variable, where its value depends on the erasure probability p_i and the flow partition. For simplicity, we let the priority factor $c_i = 1\ \forall\, i = 1, \ldots, M$. Therefore, we have an instance of the SKP corresponding to a partition of the M flows into subgroups.

Solution Reduction: Finding the solution of SKP, given the solution of CARE, is straightforward. Let us assume that $\mathcal{G}^*$ is a partition that maximizes the expected PQoS across all the users. We have that

$$\begin{aligned}\sum_{i=1}^{|\mathcal{G}^*|}\sum_{j=1}^{M_i}\mathbb{E}[\gamma_{ij}] &\overset{(a)}{\geq} \sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i}\mathbb{E}[\gamma_{ij}] \quad \forall\,\mathcal{G}\in\Omega\\ &\overset{(b)}{\geq} \sum_{i=1}^{M} x_i v_i \quad \forall\, v_i \in V\\ x_i = \{0,1\}, &\sum_{i=1}^{M} x_i w_i \leq N. \end{aligned} \tag{23}$$

The inequality (a) follows from the assumption that $\mathcal{G}^*$ is the solution to CARE, i.e., the optimal one that maximizes the expected PQoS. The inequality (b) immediately holds because the right-hand side of (b) is equivalent to that of (a), representing in the context of SKP, i.e., the value distribution of the items. Thus, the union of the subsets of $\mathcal{G}^*$ forms the subsets of items selected for the SKP, where the total value that is maximized with the weight is less than the knapsack capacity.

D. Context-Aware Partial Interflow Network Coding

We next discuss PCARE, an extension of CARE, to further improve the system PQoS. We want to emphasize that PCARE has different formulation in comparison with CARE. On one hand, CARE seeks for optimal mixing and scheduling of incoming data flows, whereby a flow (as a whole) can be either combined with other flows or transmitted separately. On the other hand, PCARE allows incoming data flows to be further divided into subflows, which consequentially are combined with other subflows (of other flows) for transmission. By doing so, PCARE can achieve a finer grained bandwidth allocation compared with CARE, albeit a more sophisticated computation. Fig. 4 illustrates an example of subflow partitioning and mixing of PCARE. In this example, we consider three incoming data flows with their corresponding FEC codes being $(n_1, k_1) = (3, 2)$, $(n_2, k_2) = (1, 1)$, and $(n_3, k_3) = (4, 3)$, respectively. We recall that an FEC code (n, k) implies that n time slots are used to deliver k data packets, where receiving any k out of n transmitted packets is sufficient to recover the data. The PCARE scheme is performed via the following steps.

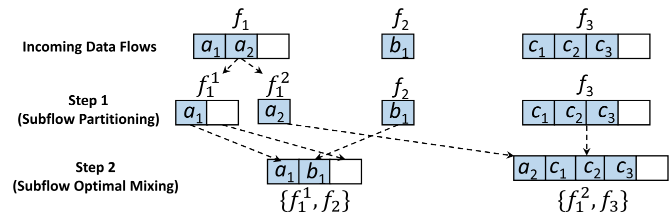


Fig. 4. Example of subflow partitioning and mixing of PCARE.

- *Subflow Partitioning*: PCARE first divides incoming data flows into subflows. We emphasize that there are many ways to divide a flow $f_i : (n_i, k_i)$ into subflows. Let $\mathbf{f}$ be the set representing all subflows of f_i. Consider a configuration where f_i is divided into J subflows $f_i^j, j = 1, \ldots, J$. We have the following constraints:

$$\mathbf{f} \supset f_i^j : \left(n_i^j, k_i^j\right), j = 1, \ldots, J$$

$$n_i = \sum_{j=1}^{J} n_i^j, k_i = \sum_{j=1}^{J} k_i^j.$$

  In Fig. 4, flow f_1 is divided into $J = 2$ subflows $f_1^1 : (n_1^1, k_1^1) = (2, 1)$ and $f_1^2 : (n_1^2, k_1^2) = (1, 1)$, while we keep flows f_2 and f_3 intact (a flow is a subflow of itself). As shown, even with this simple example, there are many ways to divide the incoming flows into subflows. Our example in Fig. 4 shows only one specific subflow configuration.
- *Subflow Optimal Mixing*: Next, given a set of subflows, they are then optimally mixed together for transmission. In our example in Fig. 4, subflow f_1^1 is combined with flow f_2, whereas subflow f_1^2 is combined with flow f_3. Data packets of each subgroup are then combined together within that subgroup for transmission. The key idea of PCARE is to divide the incoming flows into subflows, enabling finer grained bandwidth allocation for each subgroup. To illustrate this point, we next show a simple numerical example to illustrate the benefit of subflow partitioning of the PCARE scheme.

Numerical Example: Assume that $p_1 = 0.06$, $p_2 = 0.2$, and $p_3 = 0.05$ are the packet loss rates of the channels to receivers

D_1, D_2, and D_3, respectively. Supposing that we use RNC for combining data within each subgroup, we then have the probability that all receivers can recover their desired data using the PCARE scheme in Fig. 4, which is computed as follows.

- *Receiver* D_1: The probability that receiver D_1 can recover its desired data is computed by the product of the probabilities that data in each subgroup can be recovered successfully. In our example in Fig. 4, there are two subgroups, which have two and four data packets being transmitted in three and five time slots, respectively. The probability that D_1 can recover its data is computed as

$$P_1 = \sum_{i=2}^{3} \binom{3}{i} (1-p_1)^i p_1^{3-i} \times \sum_{i=4}^{5} \binom{5}{i} (1-p_1)^i p_1^{5-i}. \quad (24)$$

The first and second terms represent the probability that D_1 successfully receives data packets a_1 and a_2 in the first and second subgroups, respectively.
- *Receivers* D_2 *and* D_3: Similarly, we can compute the probabilities that receivers D_2 and D_3 can recover their desired data as

$$P_2 = \sum_{i=2}^{3} \binom{3}{i} (1-p_2)^i p_2^{3-i} \quad (25)$$

$$P_3 = \sum_{i=4}^{5} \binom{5}{i} (1-p_3)^i p_3^{5-i}. \quad (26)$$

Substituting the values of $p_1 = 0.06$, $p_2 = 0.2$, and $p_3 = 0.05$ to (24)–(26), we obtain $P_1 = 0.9581$, $P_2 = 0.8960$, and $P_3 = 0.9974$. As a result, the probability that all receivers can recover their desired data is $P = P_1 \times P_2 \times P_3 = 0.8391$.

On the other hand, the best solution CARE can be achieved by combining flows 1 and 3 together, while transmitting flow 2 separately. Using this scheme, all receivers can recover their desired data with a probability $P = 0.792$, which is much lower than that of the PCARE scheme earlier. Therefore, we can see that, by dividing traffic flows into subflows, PCARE achieves better bandwidth allocation, resulting in higher system performance. However, one can also see that finding the optimal solution to the PCARE scheme is extremely computationally expensive. Its detailed theoretical analysis is considered as our future investigation.

V. Approximation Algorithm to Context-Aware Interflow Network Coding and Scheduling

Here, we describe a simple but efficient heuristic algorithm based on the well-known MCMC method [46] to find a near-optimal solution. Although MCMC-based techniques have been extensively used to solve hard optimization problems, it is very challenging to devise an efficient algorithm to approximate the solution of a given specific problem. Typically, a system designer needs to 1) design the target distribution that reflects the solution and 2) effectively generate samples from this target distribution. Unfortunately, achieving such design goals is challenging because it is very difficult to know the stochastic properties of the system. Additionally, it is not trivial to generate samples from an arbitrary distribution and design mechanism to transition among the states. Furthermore, the convergence time of the proposed algorithm needs to be upper bounded to ensure the performance guarantee. Here, we will describe how to obtain such objectives. We start with the description of the MCMC-based approximation algorithm and then prove its upper bounded convergence time.

A. *MCMC-Based Algorithm (CARE-SAB)*

Here, we show how to appropriately construct a target distribution and use MCMC to obtain the solution. Consider a scenario with M concurrent flows traversing through the base station. Let Ω be the set of all possible partition policies and $S(\pi)$ be the average satisfaction factor of a partition policy $\pi \in \Omega$. We represent each partition policy by an M-tuple group index as $\pi = (i, j, \ldots, k)$, where i indicates that the first flow belongs to group i, the second flow belongs to group j, and so on. The objective is to maximize the average PQoS over all users. That is

$$\max_{\pi \in \Omega} S(\pi) = \max_{\pi \in \Omega} \left\{ \sum_{i=1}^{|\pi|} \sum_{j=1}^{M_i} c_{ij} \gamma_{ij} \right\}. \quad (27)$$

We should note that the size of Ω, i.e., the number of ways that M flows can be partitioned into subgroups, is very large. Based on the result of [47], we have that

$$|\Omega| = \sum_{j=1}^{M} \frac{1}{j!} \sum_{i=0}^{j} (-1)^i \binom{j}{i} (j-i)^M. \quad (28)$$

Hence, using exhaustive search, even for a reasonably small number of flows, is infeasible for time-sensitive applications. Moreover, every time a flow joins, terminates, or its channel condition changes, the AP needs to repartition again. Instead, by using the MCMC method, we will show that the time to achieve the near-optimal solution will be substantially reduced.

We first define the target distribution as follows:

$$f(\pi) = C e^{\frac{S(\pi)}{T_B}} \quad (29)$$

where C is a normalization factor, and T_B is a "cooling" parameter that controls the process convergence. In particular, when T_B reaches to a sufficient small value, the algorithm terminates, and the best accepted configuration is returned as the solution. As shown in (29), when $S(\pi)$ increases, $f(\pi)$ also increases. Therefore, with high probability, we will draw samples corresponding to $S(\pi)$, which, by design, will maximize the average user PQoS. Next, we design a mechanism for moving from one state to another in the chain. To do so, we define a neighbor of a partition in the sample space Ω as follows.

Definition 5.1: A Partition Policy π_j is Called a Neighbor of a Partition Policy π_i iff π_i and π_j Differ in Only One Element: From the aforementioned definition, π_j can be generated from π_i by replacing an element of π_i with an element drawn randomly from the index set $\mathcal{I} = \{1, 2, \ldots, M\}$. For example, when $M = 5$, partition $\pi_i = (1, 1, 3, 2, 3)$ has a neighbor $\pi_j = (1, 1, 1, 2, 3)$ (because π_i and π_j differ only in the third element).

One should note that, in the context of the MCMC, each partition policy corresponds to a state. We now propose a simulated-annealing-based algorithm to generate samples according to the designed target distribution. We propose a transition function $q(\pi_i, \pi_j)$ that specifies the probability to move from state π_i to one of its neighboring states π_j. Specifically, an element of π_i is selected uniformly at random, and then it is replaced by one of the possible indexes uniformly. Therefore, we have that

$$q(\pi_i, \pi_j) = q(\pi_j, \pi_i) = \frac{1}{2M(M-1)}. \tag{30}$$

Consequently, the acceptance probability, i.e., the probability that the chain moves from the current state π_i to a neighboring state π_j, is given by

$$\begin{aligned}\alpha(\pi_i, \pi_j) &= \min\left\{1, \frac{f(\pi_j)q(\pi_j, \pi_i)}{f(\pi_i)q(\pi_i, \pi_j)}\right\} \\ &= \begin{cases} 1, & \text{if } S(\pi_j) \geq S(\pi_i) \\ e^{\frac{S(\pi_j)-S(\pi_i)}{T_B}}, & \text{if } S(\pi_j) < S(\pi_i). \end{cases}\end{aligned} \tag{31}$$

As defined, the chain will move from the current state π_i to a new state π_j with probability of one, if π_j is a better state (state has higher user satisfaction value). Otherwise, the chain will move to a new state π_j with probability of $e^{(S(\pi_j)-S(\pi_i))/T_B}$. With this design, the whole state space will be explored, if we run the algorithm sufficiently long. Furthermore, the Boltzmann distribution will become increasingly more concentrated around the global maximizer, by gradually decreasing the temperature T_B. Pseudocode of the simulated-annealing-based algorithm is described in Algorithm 1.

Algorithm 1: CARE-SAB Algorithm.

Input: M, c_i, FEC code (n_i, k_i).
Output: Optimal Flow Partition.
STEP 1: Initialize the starting state π_0 and temperature T_0. Set $n = 0$.
STEP 2: With probability 1/2, generate a new state π_j from the proposal $q(\pi_n, \pi_j)$.
STEP 3:
if $S(\pi_j) \geq S(\pi_n)$ **then**
  $\pi_{n+1} = \pi_j$
else
  $U \sim U(0,1)$ {Generate a uniform random variable.}
  if $U < \alpha(\pi_n, \pi_j) = e^{\frac{S(\pi_j)-S(\pi_n)}{T_n}}$ **then**
    $\pi_{n+1} = \pi_j$
  else
    $\pi_{n+1} = \pi_n$
  end if
end if
STEP 4: Decrease the temperature $T_{n+1} = \beta.T_n$ where $\beta < 1$, increase n by 1 and repeat from **STEP 2** until stopping condition satisfied (*).
STEP 5: Return a scheme π that produces the maximum weighted-average satisfaction factor.

Remark (*): The stopping condition can be set by the number of iterations or the difference between the two consecutive states is less than a prespecified value.

B. Upper Bound of Convergence Time

1) Convergence Correctness: The guarantee of convergence to the target distribution using the CARE-SAB algorithm is shown via the following theorem.

Theorem 5.2: Samples drawn from the CARE-SAB algorithm form a Markov chain (MC) whose states satisfy the detailed balance equation

$$\theta(\pi_i)P(\pi_i, \pi_j) = \theta(\pi_j)P(\pi_j, \pi_i) \quad \forall \pi_i, \pi_j \in \Omega \tag{32}$$

where $\theta(\pi_i)$ and $\theta(\pi_j)$ are the stationary distributions of states π_i and π_j; $P(\pi_i, \pi_j)$ and $P(\pi_j, \pi_i)$ are, respectively, the transition probabilities from state π_i to state π_j and vice versa.

Proof: The proof is provided in the Appendix. ■

The result of Theorem 5.2 shows that samples drawn from the designed algorithm form an ergodic MC, i.e., every state can go to every state; thus, it is possible to find an optimal solution as long as the algorithm runs sufficiently long.

2) Convergence Time: Here, we will derive an upper bound of convergence time to the target distribution. We first define the variation distance at time step k with respect to an initial state π_0 of the MC as

$$\triangle_{\pi_0}(k) \triangleq \max_{\pi \in \Omega} \left| P^k(\pi_0, \pi) - \theta(\pi) \right|. \tag{33}$$

Then, the convergence time of the MC to the target distribution is measured by

$$\tau_{\pi_0}(\varepsilon) \triangleq \min\{k : \triangle_{\pi_0}(k') \leq \varepsilon \quad \forall k' \geq k\} \tag{34}$$

where $0 < \varepsilon$ is an arbitrary infinitesimal value specifying how close the desired solution to an optimal solution. To bound the convergence time at which the chain approaches its stationary distribution (optimal solution), we use the canonical path technique [21]. Letting $e = (\pi_x, \pi_y) \in \Omega^2$, we define $Q(e) \triangleq Q(\pi_x, \pi_y) = \theta(\pi_x)P(\pi_x, \pi_y)$, and a graph $G = (\Omega, E)$, where $(\pi_x, \pi_y) \in E$ *iff* $Q(\pi_x, \pi_y) > 0$. For every ordered pair $(\pi_x, \pi_y) \in \Omega^2$, a canonical path ς_{xy} through G from π_x to π_y is specified by a sequence of legal transitions in G that leads from initial state π_x to final state π_y. Let $\Gamma \triangleq \{\varsigma_{xy} : \pi_x, \pi_y \in \Omega\}$ denote the set of all canonical paths. We now define the edge congestion for the set Γ as follows:

$$\rho(\Gamma) \triangleq \max_{e \in E} \frac{1}{Q(e)} \sum_{\varsigma_{xy} \ni e} \theta(\pi_x)\theta(\pi_y)|\varsigma_{xy}| \tag{35}$$

where $\gamma_{xy} \ni e$ implies that ς_{xy} uses the directed edge e, and $|\varsigma_{xy}|$ denotes the length of the path. The convergence time is bounded by the following proposition.

Proposition 5.1: Let $\mathbf{M}$ be a finite, time-reversible, and ergodic MC over Ω with self-loop probabilities $P(x,x) \geq 1/2$ for all $x \in \Omega$ and stationary distribution π. If the congestion of $\mathbf{M}$ is ρ, then the mixing time of $\mathbf{M}$ satisfies $\tau_x(\epsilon) \leq \rho(\ln \pi(x)^{-1} + \ln \epsilon^{-1})$, for any choice of initial state x.

We are now ready to bound the mixing time of the proposed approximation algorithm CARE-SAB. We choose canonical paths ς_{xy} from any state π_x to any state π_y, which takes T steps, changing π_{x_i} to π_{y_i} on the ith step. Thus, when $\pi_{x_i} = \pi_{y_i}$ for some i, the ith step is just a self-loop. We derive a bound for the mixing time of the chain. Considering any edge $e = (\pi_u, \pi_v)$ for $u \neq v$, we have

$$Q(e) = \theta(\pi_u)P(\pi_u, \pi_v) \overset{(a)}{=} Ce^{\frac{S(\pi_u)}{T_B}} \frac{\min\left\{1, e^{\frac{S(\pi_v)-S(\pi_u)}{T_B}}\right\}}{2M(M-1)}$$
$$= \frac{C}{2M(M-1)} \min\left\{e^{\frac{S(\pi_u)}{T_B}}, e^{\frac{S(\pi_v)}{T_B}}\right\} \overset{(b)}{\geq} \frac{C}{2M(M-1)} \tag{36}$$

where (a) follows from (29)–(31), and (b) follows by using the fact that a partition policy could have a zero-PQoS value. In addition, we have

$$\theta(\pi_u)\theta(\pi_v) = C^2 e^{\frac{S(\pi_u)+S(\pi_v)}{T_B}} \leq C^2 e^{\frac{2}{T_B}} \tag{37}$$

where the last inequality follows from the fact that $S(\pi) \leq 1$ for any partition scheme π. We now compute the number of canonical paths ς_{xy} that use edge e. Note that a canonical path ς_{xy} uses edge $e = (\pi_u, \pi_v)$, where π_u and π_v differ only in the ith element *iff* $\pi_{x_j} = \pi_{u_j}$ for $j = i, \ldots, T$ and $y_j = v_j$ for $j = 0, \ldots, i$. Thus, the number of canonical paths that use edge e is K_m^{T-1}. We now bound the edge congestion as

$$\rho = \max_{e \in \Omega} \frac{1}{Q(e)} \sum_{\varsigma_{xy} \ni e} \pi(x)\pi(y)|\varsigma_{xy}|$$
$$\leq 2CT^2(K_m - 1)(K_m)^{T-1} e^{\frac{2NK_m}{T_B}}. \tag{38}$$

In addition, we have $C = 1/\sum_{x\in\Omega} e^{(S(x))/T_B} \leq 1/|\Omega| = 1/K_m^T$. Therefore

$$\rho \leq 2T^2 e^{\frac{2NK_m}{T_B}}. \tag{39}$$

Using the result from Proposition 5.1 and noting that $\pi(x) \geq 1/|\Omega| = 1/K_m^T$, we then have the convergence time bounded by

$$\tau_x(\epsilon) \leq 2T^2 e^{\frac{2NK_m}{T_B}} \left(\ln K_m^T + \ln \epsilon^{-1}\right). \tag{40}$$

As expected, when the value of ϵ decreases (i.e., closer to the optimal solution), it requires longer runtime. However, as we will show in the simulation, on the order of hundred iterations, the approximation algorithm can obtain a close-optimal solution.

VI. Simulations and Discussions

A. Basic Setup

Network Parameters: We consider a realistic wireless access network having different types of applications with time-varying channel conditions. We first consider a network consisting of five data flows of different applications and compare the performance of different transmission strategies. We then increase the number of data flows to evaluate the robustness of the approximation algorithm. We assume that there are two classes of services: delay-sensitive and elastic traffic. The transmitter decides a coding scheme for a flow, based on the cost at which the user had paid to the service provider, i.e., the higher cost, the higher priority. Note that the service class and priority of a data flow can be easily elaborated in the header of the transmitted packets. In the UNI technique, the transmitter uses the priorities of the incoming flows to assign their redundancies, and they will be used in all the techniques for a fair comparison. We consider a wireless channel with a bandwidth of 2 Mb/s, which is equivalent to $N = 133$ time slots or 133 1.5-kB packets. In addition, an elastic traffic requires 18 data packets per second, corresponding to a rate of 27 kb/s, while a delay-sensitive traffic requires 25 and 30 data packets, corresponding to rates of 37.5 and 45 kb/s, for medium and high QoS, to achieve a PQoS value of one. Our parameters are set based on the number of frames per second in video streaming and the standard service specifications [48]. For example, our delay-sensitive flow can be used to model a Voice over IP (VOIP) call (e.g., using G.728 standard with a codec interval of 5 (ms) requires a transmission rate of 31.5 kb/s [48]).

TABLE I
Parameters of the Incoming Flows

Flow ID	Rx ID	FEC (n_i, k_i)	p_i	Service type	Priority
$f1$	D_1	$(21, 18)$	–	Elastic	1
$f2$	D_2	$(21, 18)$	0.05	Elastic	1
$f3$	D_3	$(22, 18)$	0.05	Elastic	2
$f4$	D_4	$(31, 25)$	0.05	Delay-sensitive	3
$f5$	D_5	$(38, 30)$	0.05	Delay-sensitive	4

If the number of data packets received at each receiver is less than the required packets, its satisfaction will decrease in accordance to the PQoS functions, as described in Section III-C. The transmission parameters of the incoming flows are given in Table I. These parameters are set based on the types of applications, priorities of the incoming flows, and the bandwidth availability. In addition, the redundancy used for each incoming flow depends on its priority; for example, in our experiments, we set priorities 1, 2, 3, and 4, corresponding to redundancies of 15%, 20%, 25%, and 30%, respectively. Note that these parameters will be applied to all techniques.

Transmission Strategies: We evaluate the performance of the following transmission strategies for comparison.

- *Unicast (UNI)*: The UNI scheme transmits data of the incoming flows separately, without using NC.
- *Random Network Coding (RNC)*: RNC scheme implements a standard interflow NC technique, where data packets of all flows are combined together using RNC [41]. The coded packets are then broadcasted to all the receivers.
- *Systematic Random Network Coding (SRNC)*: SRNC implements a simple systematic NC, where original data packets are transmitted in the first phase and coded data packets (generated by using RNC across all data flows) are transmitted in the second phase.
- *Type of Flow (ToF)*: The first naïve ToF scheme mixes all data of flows with the same application types. Such a transmission strategy does not have to perform intensive optimization computation; however, it might limit the effect of mismatched mixing of the NC-based schemes, by combining data of flows with the same types.

- *Priority of Flow (PoF) Scheme*: Differently, the PoF scheme selects flows that have the same priority as subgroups to perform NC. The idea of using the PoF scheme is to utilize the priority factors of the incoming data flows to quickly classify them into different subgroups for NC. Intuitively, encoding data of the same priority flows could enhance the QoS of the network.
- *Network Coding for Throughput (NCT)*: The NCT scheme is simulated based on the work in [5], which is proposed for maximizing the network throughput. In NCT, the sender considers the packet at the head of its queue as a primary packet and selects side packets to construct coded packets so that it maximizes the number of possible receivers at the current time slot.
- *CARE/CARE-EXH*: This scheme uses the optimal mixing solution via exhaustive search for data transmission.
- *CARE-SAB*: CARE-SAB uses an approximation algorithm based on the proposed CARE-SAB algorithm for data encoding and scheduling.
- *2-PCARE*: Based on the PCARE scheme described in Section IV-D, we simulate 2-PCARE for comparison. In this scheme, each incoming flow is divided into two equal subflows, and then, these subflows are optimally combined for transmission.

B. Data Recovery

We first show the benefit of *informed* mixing and the drawback of blind mixing by examining the probability that all the receivers can decode their packets using strategies derived in Section IV. Fig. 5(a) shows the probability of data recovery versus partition policies, i.e., the way of mixing data when packet losses of receivers from D_2 to D_5 are set to 5% while that of receiver D_1 is 13%. We map each partition policy to an integer on the x-axis. The number of possible partition policies is an exponential function of M, and this is equal to the sum of the Stirling numbers of the second kind, as shown in (28). We also plot the recoverability probabilities for UNI and SRNC techniques on the same graph for comparison. They are indicated by straight lines since these techniques do not depend on the partition policies. Recall that UNI does not mix packets from different flows. SRNC sends the original packets and then the mixed redundant packets; hence, the amount of mixing here is rather minimal. As observed, SRNC is clearly better than UNI. It is interesting to note that, at least in this scenario, blind mixing is generally better than UNI. As expected, CARE-SAB results in different recoverability probabilities, depending on which flows are combined with each other. In our program, we let the CARE-SAB algorithm run until converged. Based on our deeper data analysis generated by the program, the proposed CARE-SAB finds the best partition by mixing flows f_1 and f_4 into one group and flows f_2, f_3, and f_5 into another group.

Next, we evaluate the data recovery probability by all receivers versus the packet loss probability in Fig. 5(b). In this scenario, the packet loss probabilities of receivers $D_i, i = \{2, \ldots, 5\}$ are shown in Table I, while the packet loss rate of receiver D_1 is varied from 1% to 22%. As expected, CARE-SAB outperforms the other schemes, with considerable gaps,

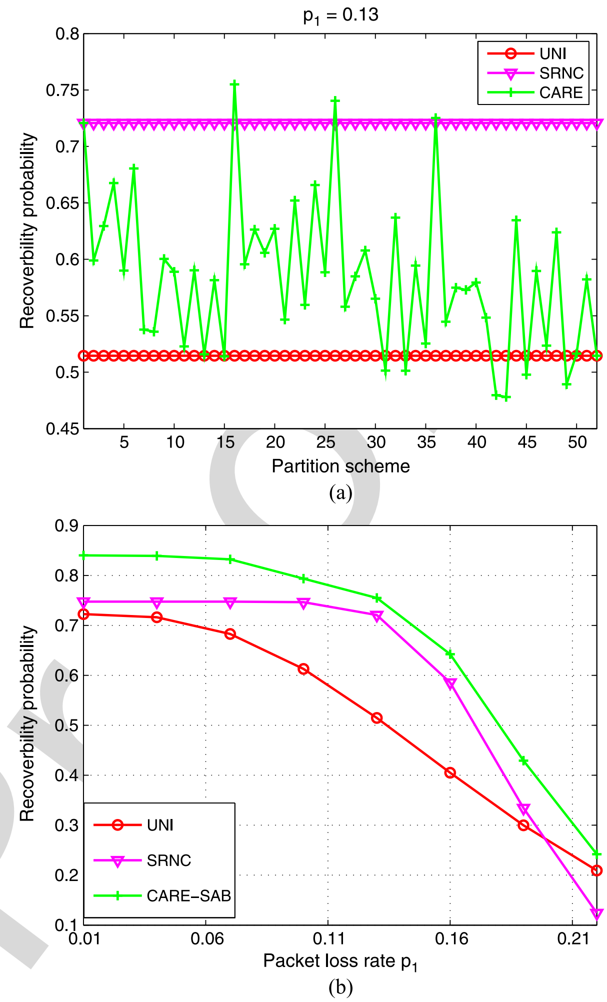


Fig. 5. Recoverability probability versus (a) partition scheme and (b) packet loss rate p_1.

due to its selective mixing of the flows. In addition, we observe that, when p_1 is less than 18%, SRNC achieves better performance than UNI. In other words, in this erasure regime, mixing data packets across all flows would be more beneficial than transmitting them separately. However, this is not the case when the packet loss p_1 is greater than 18%. In such a setting, the UNI scheme outperforms SRNC. The intuition is that SRNC combines the data of all the flows; as a result, each receiver needs to obtain at least a full set of coded packets to recover its data. However, this may not be possible to receiver D_1 because it experiences a deep fading, leading to substantial reduction on the overall network recoverability. In such a case, separately transmitting data to different users will be a better option.

C. User's PQoS Versus Erasure Probability

We first evaluate the individual PQoS versus the transmission channel conditions. In particular, we set $p_3 = p_4 = 5\%$, $p_2 = p_1 + 0.01$, and $p_5 = p_1 + 0.02$. The other parameters of the network are set the same as before in Table I. The base values of the PQoS, i.e., γ_0, of the delay-sensitive and elastic traffic are set at 0.5 and 0.6, when the number of useful packets received equals 50% of the intended packets. This setting is to reflect that delay-sensitive applications are more vulnerable to

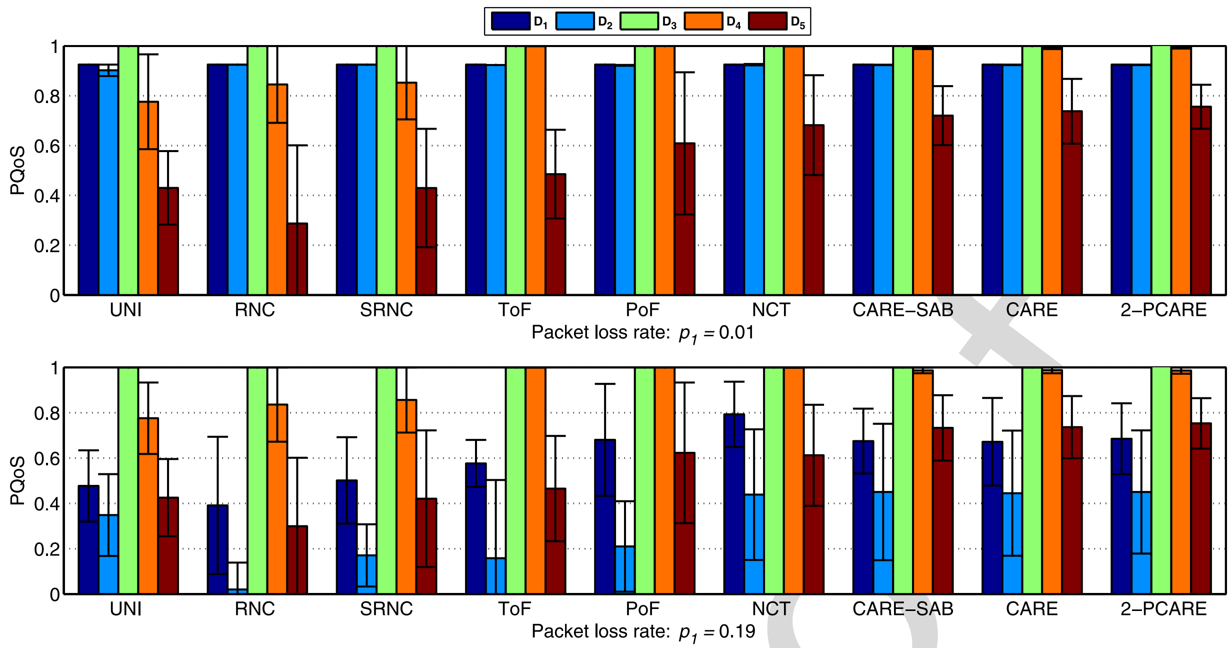


Fig. 6. PQoS values of users in different packet error regimes (order of the receivers is the same, as illustrated in the legend).

packet losses than the elastic traffic. The CARE-SAB algorithm runs for 30 iterations, using the normalization factor $C = 1$, initial temperature $T_0 = 1$, and cooling scale factor $\beta = 0.9$. We use the simplest partition policy $k = 2$ for the PCARE scheme, where each flow is evenly divided into two subflows. We perform many trials and compute the average of the results.

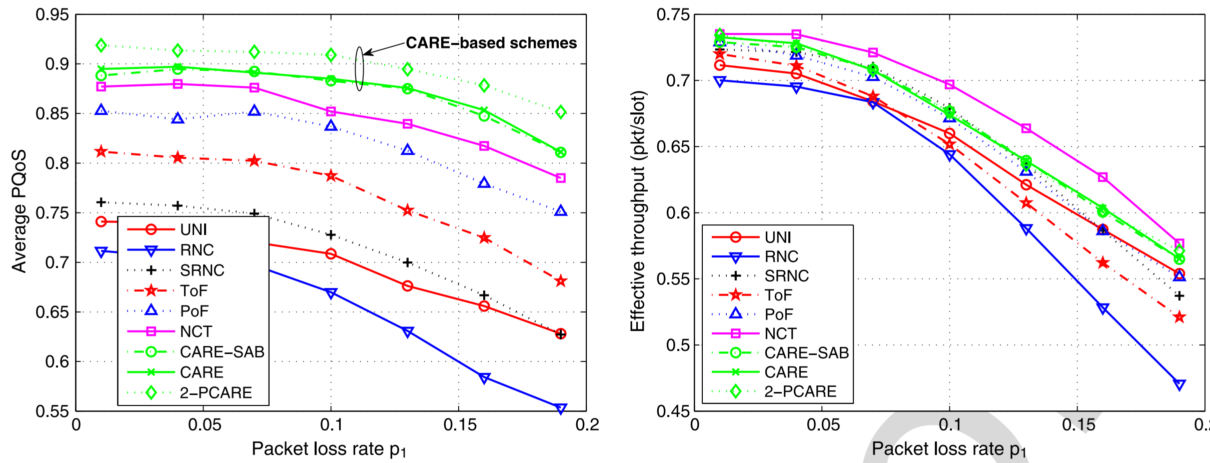
Fig. 6 represents the PQoS values across all users in two different regimes of the packet erasure probabilities. In the low-packet-loss regime (upper part in Fig. 6), i.e., $p_1 = 0.01$, all the strategies satisfy the QoS of the first four flows. This is intuitively plausible since, in this case, transmission bandwidth is plenty for these flows, no optimization is needed, and all these users get what they want. However, the RNC scheme significantly decreases the PQoS of D_5. This is because, when combining all the data packets together, it cannot recover the transmitted data, resulting in a degraded PQoS. On the other hand, the CARE-based schemes that balance data types and priorities for different groups achieve the best performance. Furthermore, the 2-PCARE scheme with a finer grained data partition policy achieves the best performance.

On the other hand, in the high-packet-loss regime (lower part in Fig. 6), i.e., $p_1 = 0.19$, all the receivers with low packet loss rates, i.e., D_3 and D_4, can still maintain a high PQoS in all transmission strategies. However, the PQoS of the receivers with higher packet loss rates, i.e., D_1, D_2, and D_5, significantly decreases in all strategies. In particular, receiver D_2 has its PQoS decreased substantially in the SRNC scheme. The intuition is that mixing up data of all flows makes it difficult for D_2 to receive a full set of the coded packets in the high-erasure-probability regime. As a result, D_2 is not able to recover its data. As expected, CARE (i.e., exhaustive search) that searches all the possibilities of partition policies always achieves the best performance. However, an interesting observation is that the CARE-SAB algorithm can approximate the optimal solution with only 30 iterations. This significantly reduces the search time compared with the exhaustive search in larger size problems. Indeed, the PQoS achieved by CARE-SAB is very close to that of the exhaustive search CARE, with a marginal gap.

D. Network PQoS and Effective Throughput Versus Erasure Probability

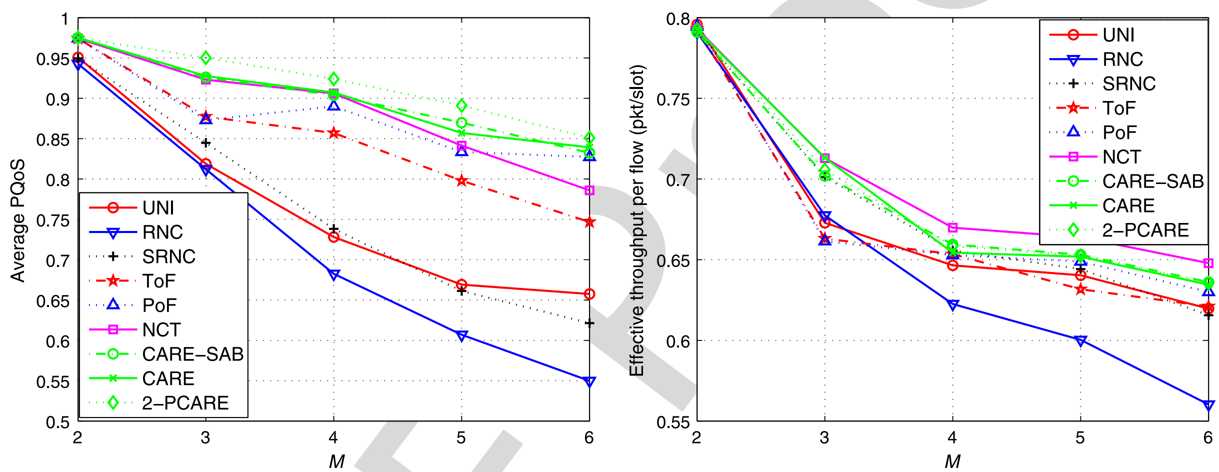
Next, we compare the network PQoS and effective throughput of different transmission strategies versus the packet loss p_1. The effective throughput is computed based only on the received data, contributing to the QoS of the users, without considering application types. Fig. 7(a) shows the average PQoS across all receivers. As expected, the average PQoS decreases with the increase of p_1. We observe that RNC has the worst PQoS out of all strategies. This is because of the degraded PQoS of receivers that cannot recover the transmitted data due to bad channel conditions. Interestingly, UNI outperforms RNC with a significant performance gap, due to its partially recovered data. On the other hand, in RNC, the receivers need to receive a full set of coded packets for data decoding; however, this may not be possible due to poor transmission channel conditions. The NCT outperforms the other schemes but less than that of the CARE-based schemes, as shown in Fig. 7(a). This is because greedily optimizing the throughput without considering the characteristics of the traffic could significantly decrease the network QoS. Again, 2-PCARE achieves the best performance, which is followed by CARE-based schemes. We further observe that CARE-SAB achieves an identical performance of CARE (i.e., exhaustive search) with much less runtime. This confirms the efficiency and robustness of the proposed CARE-SAB algorithm.

Next, we investigate the network effective throughput versus the packet error rate in Fig. 7(b). As expected, NCT achieves the best performance because its objective is to maximize the



(a) (b)

Fig. 7. Network performance of different schemes versus p_1 with $M = 5$: (a) average PQoS and (b) effective network throughput.



(a) (b)

Fig. 8. Network performance of different schemes versus M for (a) average PQoS and (b) effective network throughput.

network throughput. Interestingly, CARE-based schemes not only outperform the other techniques but also approximate to NCT, despite that the objective of CARE is to maximize the average PQoS. The explanation is as follows. First, CARE uses PQoS in its formulation, whereby the value of PQoS is computed based on both the effective throughput (i.e., number of useful packets) and characteristics of the information flows. Second, and more importantly, PQoS functions are well designed, so that a small change in effective throughput is transformed into the users' PQoS. Thus, optimizing the PQoS results in a near-optimum network effective throughput. We additionally observe that the RNC scheme suffers from combining the data of all flows, resulting in the worst performance, particularly in the presence of deep channel fading.

E. PQoS and Throughput Versus Number of Flows

We next examine the overall network PQoS and effective throughput versus the number of data flows in Fig. 8(a) and (b). As expected, the CARE-based schemes outperform the others with considerable gaps. The RNC scheme achieves the worst performance and significantly decreases as M increases. This is because all-flow encoding suffers from the curse of "all or nothing" of the RNC decoding constraint, i.e., it requires a full set of encoded packets for data recovery. SRNC outperforms UNI in the regime of lower packet error rate, while significantly decreasing with the increase of p_1. This is because encoded packets cannot be recovered due to high lost packets. Interestingly, the two heuristic ToF and PoF schemes achieve better performance compared with the traditional NC-based and UNI schemes. However, when M increases, their performance starts decreasing considerably. As expected, the 2-PCARE scheme achieves the best performance, due to its finer grained subflow partition and encoding. We also observe that the CARE-SAB obtains a competitive performance by using only 30 iterations.

Additionally, Fig. 8(b) illustrates the network effective throughput versus M. As expected, the NCT that is optimized for the throughput achieves the best performance. CARE-based schemes outperform the other techniques, despite that the objective of CARE is to maximize PQoS. This is because PQoS is constructed from both the effective throughput (i.e., number of useful packets) and characteristics of the information flows. Thus, optimizing the PQoS will result in high network effective throughput. The RNC achieves the worst network throughput,

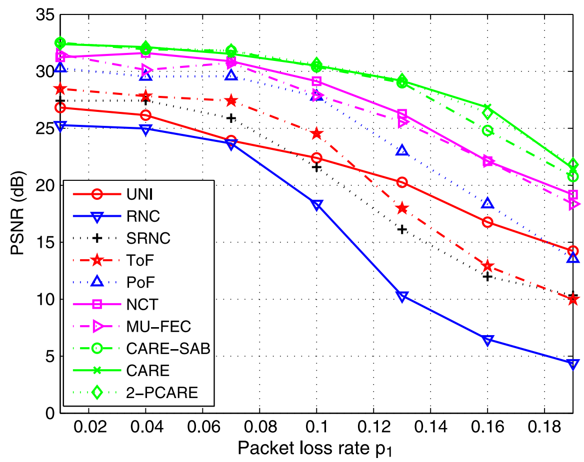


Fig. 9. Average PSNR of video sequences at receivers versus packet lost rate p_1.

and its performance decreases significantly with the increase of M. This is consistent with the intuition that when M increases it is more difficult for the receivers to receive a full set of encoded data for decoding.

F. PSNR Versus Erasure Probability

In addition, we also evaluate the average PSNR based on the luminance (Y) component of *Foreman* and *Coastguard* video sequences. We assume that each frame of the video sequences is packetized into an independent network abstraction layer of 1500 bytes. Furthermore, in our simulation, we use short video sequences, with the *Foreman* and *Coastguard* having 30 and 25 packets, respectively. The longer video sequences can be constructed by concatenating multiple frames. Additionally, we assume that the PSNR of the encoded sequences *Foreman* and *Coastguard* are 29.95 dB [15] and 45.72 dB [6], respectively, corresponding to no-error transmission of those streams.

Fig. 9 illustrates the average PSNR of the two video sequences using different transmission strategies. In our simulation, the transmitter transmits five data flows, consisting of three elastic and two delay-sensitive flows, to five receivers. To compute PSNR, we extract only the received packets of the two video sequences. As expected, when the packet lost rate is small, all schemes perform well with high PSNR. This is because only some of the transmitted data packets were lost. However, when the packet lost rate increases, the video quality at the receivers rapidly degrades. As expected, the proposed CARE-based strategies achieve the highest video qualities, due to their content-aware encoding and scheduling. We also simulated the MU-FEC scheme proposed in [49] for comparison. The MU-FEC scheme exploits intra- and interflow NC for mixing data of different flows at the transmitter to improve bandwidth efficiency. In spite of optimizing its coding for maximizing the network throughput, it does not consider the content of the traffic, resulting in low PSRN. This is the same observation for the NCT scheme, which focuses on throughput maximization instead of network QoS. Interestingly, we observe that the heuristic PoF scheme achieves very good performance in terms of PSNR by combining data of only higher priority flows associated with the video sequences. In the regime of high erasure rate, with a smaller encoding data batch, it can successfully transmit data to the receivers with higher probability. On the other hand, RNC combining all data together for transmission suffers because most of the data cannot be recovered at the receivers.

G. CARE-SAB Convergence Rate

We now evaluate the effectiveness and robustness of the CARE-SAB algorithm. In these experiments, we consider data traffic consisting of eight data flows, where the first five flows are the same as before (in Table I), and the additional flows include one elastic and two delay-sensitive flows that are a copy of D_2 and D_4, respectively. We have a total of more than 4×10^3 possible flow partition policies. To illustrate the convergence of the proposed CARE-SAB, we vary the channel conditions randomly with the packet loss rates in the range between 1% and 20%. In the CARE-SAB algorithm, the initial state is set by grouping all flows together. Fig. 10(a) and (b) illustrates a snapshot of the PQoS values changing with respect to the iteration and actual runtime, respectively. The CARE-EXH achieves the best performance by using exhaustive search for all possible partition policies and selects the best partition. On the other hand, CARE-SAB schemes implement the proposed approximation method to find the optimal partition policy. The CARE-SAB-Iter represents state by state the chain visits in each iteration, whereas the CARE-SAB records the maximum PQoS values, which have been obtained up to that iteration. The CARE-SAB schemes first aggressively "explore" states, even the ones with low value of PQoS, and gradually "cool" down to the optimal solution. As illustrated in Fig. 10(a), it takes about 200 iterations (about 5% of the search space) for the CARE-SAB schemes to "hit" the optimal solution, indicated by the PQoS merged to that of the CARE-EXH. The simulation result is consistent with the theoretical analysis of the Theorem 5.2, i.e., setting the number of iterations sufficiently large, an arbitrarily near-optimal solution can be obtained via the proposed CARE-SAB algorithm.

We further measure the actual runtime of the algorithm based on the elapsed time of the algorithm implemented on our laptop (OS Window 7, Intel Core i5 with 4-GB RAM). As illustrated in Fig. 10(b), it takes about 0.46 s for the CARE-SAB schemes to find the optimal solution for the case of $M = 8$ flows. We note that this is only one snapshot of a trial. In practice, the runtime could be much less than that, if we eliminate the effect of other concurrent processes in the machine.

We further evaluate the convergence robustness of the proposed CARE-SAB by using different values of β. We recall that β is the parameter controlling the "cooling" process of the search algorithm. Fig. 11(a) illustrates the convergence of CARE-SAB-Iter for different values of β in [0.9, 0.99]. As observed, it requires more time for the system with smaller values of β to converge to the optimal solution. This is because the system with smaller values of β (i.e., $\beta \times T_n$ decreases faster) "jumps" with larger steps at the beginning of the process that may visit several "bad" states before converging to the optimal solution. On the other hand, with greater values of

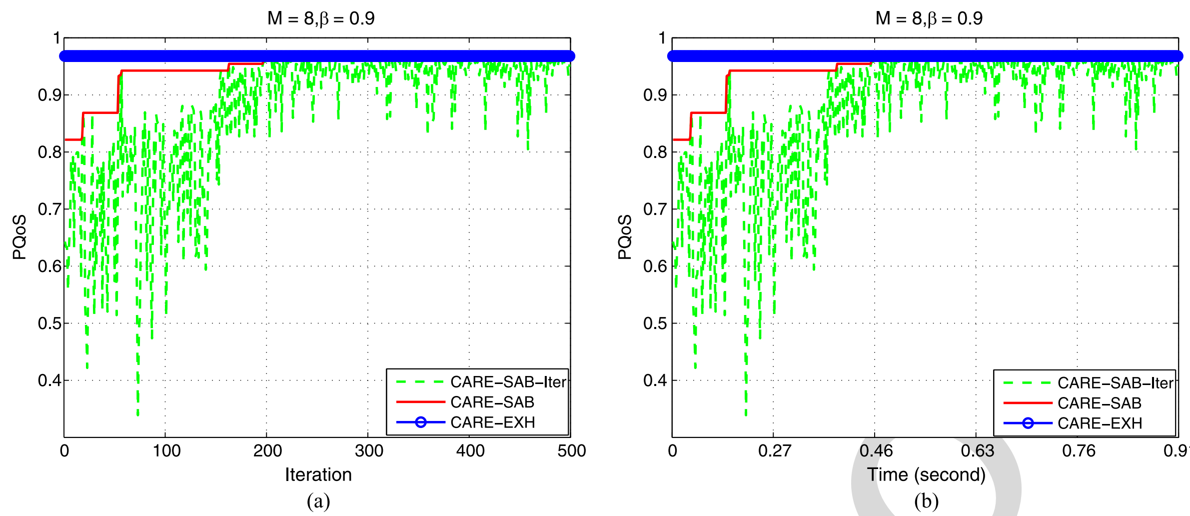


Fig. 10. CARE-SAB convergence versus (a) iteration and (b) time (in seconds): $M = 8$, $T_0 = 100$, and $\beta = 0.9$.

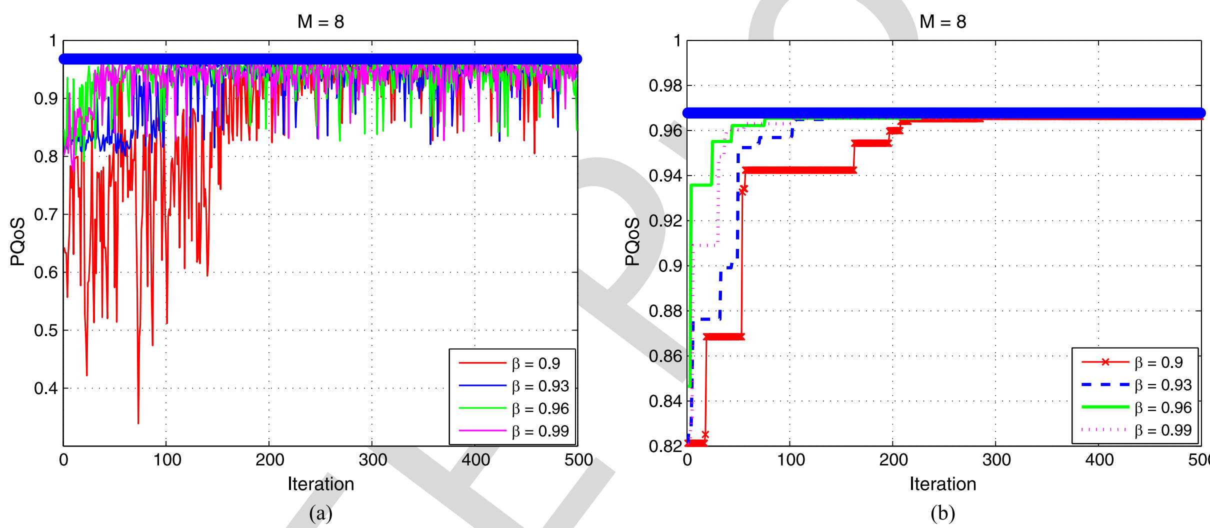


Fig. 11. CARE-SAB convergence for different values of β, $M = 8$, $T_0 = 100$.

β, the system is more conservative in giving high probability for exploring "good" states. This setting could reduce the "burning" time of the process before convergence, but it may result in a local optimal solution. Our experiments illustrated in Fig. 11(a) shows that the system quickly converges to the optimal solution within 300 iterations, for all implemented values of β. Fig. 11(b) illustrates the maximal value of PQoS that has been found up to the current iteration. Additionally, the result shows that, for some case, e.g., $\beta = 0.99$, the algorithm quickly obtains the optimal solution in only 100 iterations (about 2.5% of the search space).

Similar performance is obtained for the case of $M = 10$, as illustrated in Fig. 12. As we can observe, the proposed algorithm CARE-SAB quickly converges to the optimal solution and is robust with respect to the values of β. For some cases, e.g., $\beta = 0.9$, the CARE-SAB needs more iterations to find the optimal solution (about 450 iterations). However, we should note that, when $M = 10$, there could have 115 975 possible partition policies. Therefore, the increase is justifiable compared with the exponential increase of the search space.

Additionally, we further evaluate the convergence rate of the proposed CARE-SAB with different values of the initial temperature T_0. Fig. 13(a) and (b) illustrates the convergence of CARE-SAB-Iter and CARE-SAB for $M = 8$ and $\beta = 0.9$, respectively. As illustrated, the proposed algorithm is robust with respect to different initial values of T_0. We further observe that initializing T_0 with different values only slightly affects the system convergence rate. This is an expected result and consistent with the theoretical analysis because the temperature controls how the algorithm explores the search space. In particular, greater value of T_0 provides more freedom to the algorithm to explore more states, resulting in longer convergence time. However, such a setting will ensure that the global optimal solution will be "hit" with high probability. Similar results are also obtained for the case of $M = 10$ flows, as illustrated in Fig. 14. With larger search space, in the worst case ($T_0 = 100$), the algorithm requires about 400 iterations to find the optimal solution.

Finally, we evaluate the scalability and efficiency of the proposed CARE-SAB in Fig. 15. In particular, Fig. 15(a) compares the performance of the CARE-SAB using different

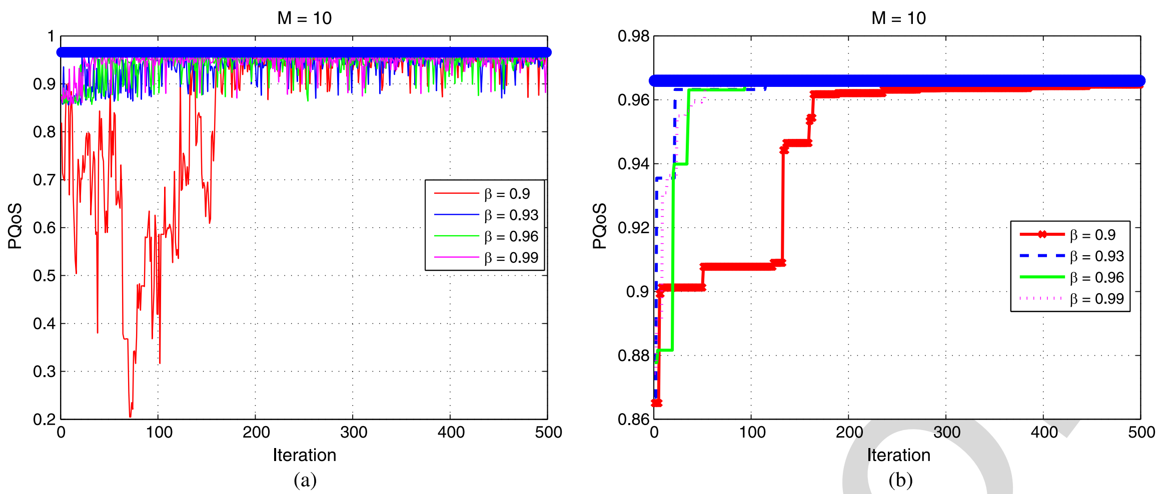


Fig. 12. CARE-SAB convergence for different values of β, $M = 10$, $T_0 = 100$.

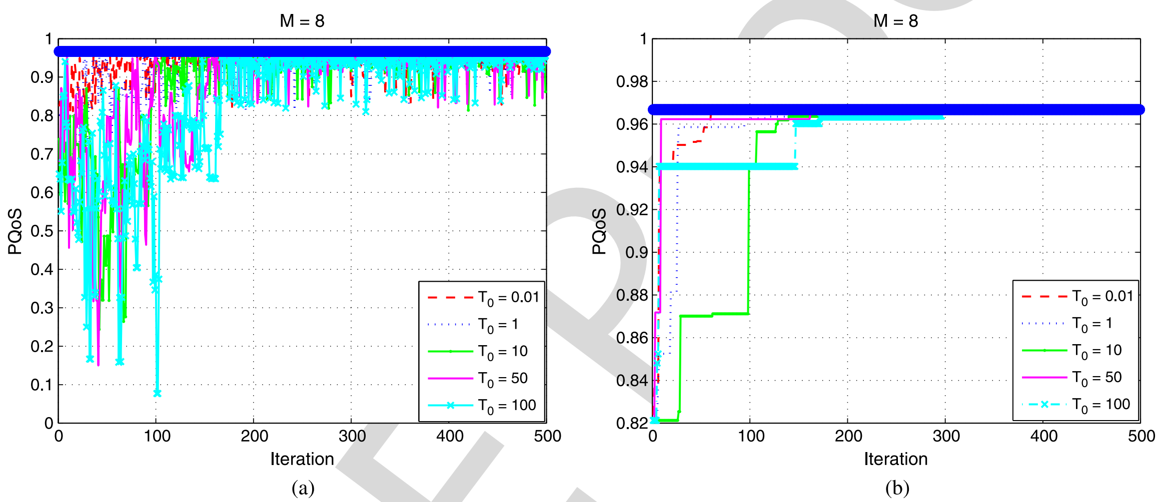


Fig. 13. CARE-SAB convergence for different values of T_0, $M = 8$, $\beta = 0.9$.

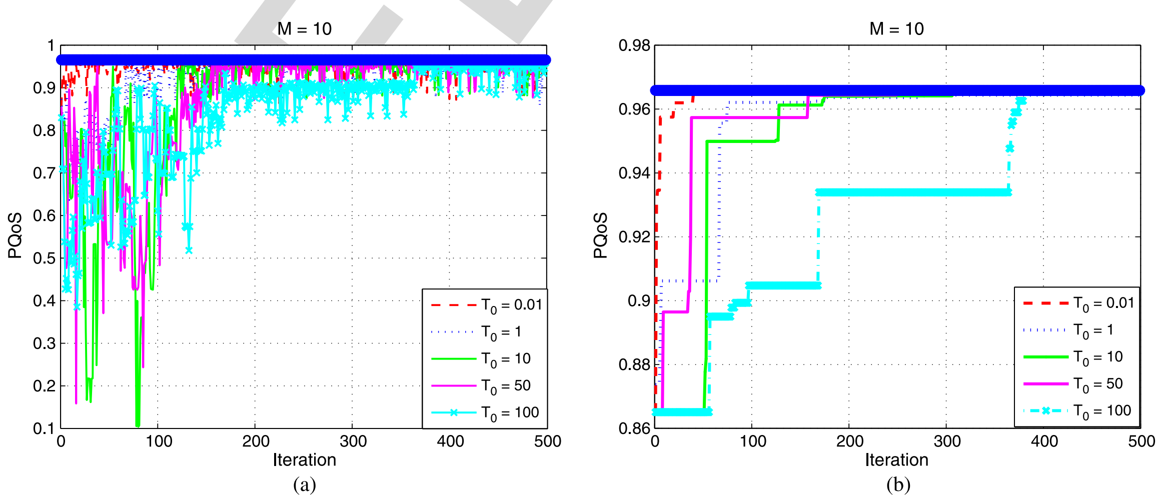


Fig. 14. CARE-SAB convergence for different values of T_0, $M = 10$, $\beta = 0.9$.

values of iterations with the exhaustive search CARE-EXH versus M, where CARE-10 and CARE-50 represent CARE-SAB that uses 10 and 50 iterations, respectively. As expected, when M increases and given a fixed number of iterations, the performance of CARE-SAB schemes decreases due to the larger search space. However, it is interesting to observe that CARE-SAB schemes can achieve about 90% performance of the exhaustive search despite using a fixed number of iterations. The results illustrate that the designed algorithm is scaled well with the problem size.

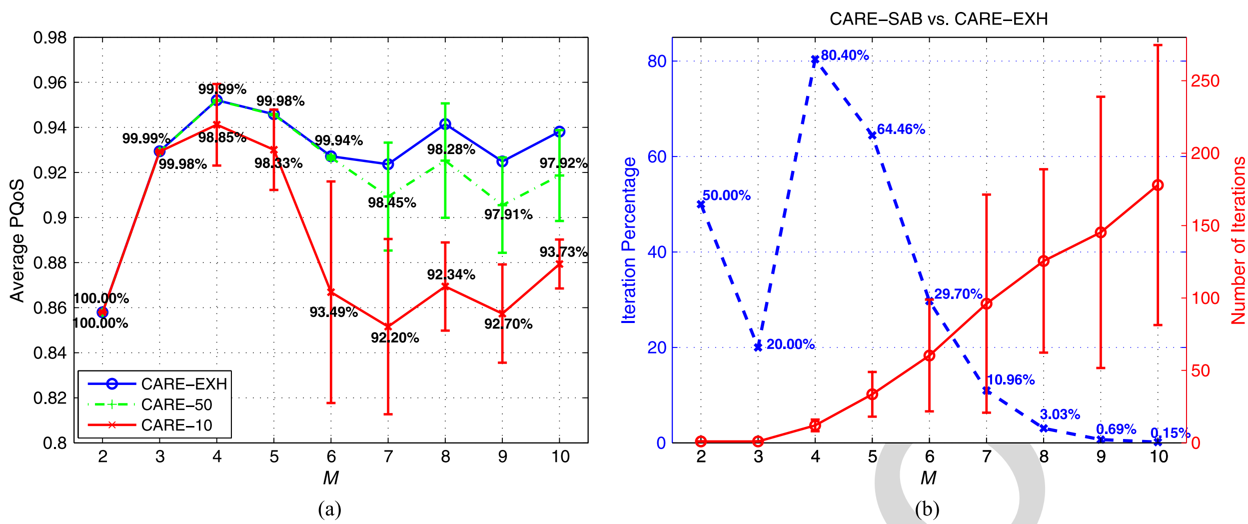


Fig. 15. Evaluating the scalability and robustness of CARE-SAB with $T_0 = 1$ and $\beta = 0.9$: (a) average PQoS for different numbers of iterations and (b) number of iterations for CARE-SAB to obtain optimal partition policy and percentage compared with CARE-EXH.

We further evaluate the scalability of CARE-SAB algorithm by changing M in Fig. 15(b). In particular, we compute the average number of iterations that the CARE-SAB algorithm consumes to find an optimal solution. In our experiment, for each M, we run the algorithm many times and compute the average of the results. Fig. 15 illustrates the average number of iterations of CARE-SAB and the corresponding percentage compared with CARE-EXH for $M = 2, \ldots, 10$. As expected, the number of iterations of CARE-SAB (i.e., the red line) increases with M, due to the exponential increase of the search space. However, compared with the exhaustive search CARE-EXH, the number of states that CARE-SAB explores to find the optimal solution decreases significantly (i.e., the dash blue curve), i.e., from 50% for $M = 2$ to 3.03% for $M = 8$ and to 0.15% for $M = 10$. The simulation results also confirm that the proposed CARE-SAB algorithm is also scaled well with M.

VII. Conclusion

We have investigated the problem of mixing data of traffic with different service classes to improve the network QoS. We first showed that exhaustively mixing data across different data flows at every opportunity may substantially decrease the network QoS. We then proposed CARE, a context-aware interflow network coding and scheduling, to maximize the QoS across all the receivers based on the user satisfaction PQoS. The objective function of CARE is formulated by considering not only the characteristics of traffic but also the service classes and channel conditions. We then showed the hardness of finding an optimal solution to CARE and proposed an efficient approximation algorithm, i.e., CARE-SAB, to obtain a guaranteed near-optimal solution. We further proved the correctness and derived an upper bound on the convergence time of the CARE-SAB algorithm. In addition, we described the PCARE scheme that partially combines data of different flows to further improve the network performance. Simulation results showed that up to a 50% performance gain of the proposed CARE-based schemes can be achieved compared with the existing approaches (e.g., RNC). The results also showed that the approximation algorithm is robust with respect to the heuristic parameters and well scaled with the number of data flows. To the best of our knowledge, this work is one of a few works studying NC from the QoS point of view. One of our future extensions is to investigate an efficient algorithm for optimal subflow partitioning and encoding of the PCARE scheme.

Appendix
Proof of Theorem 5.2

Let us consider any two states $\pi_i, \pi_j \in \Omega$. According to the proposed target distribution, we have $\theta(\pi_i) = Ce^{(S(\pi_i))/T_B}$ and $\theta(\pi_j) = Ce^{(S(\pi_j))/T_B}$. Therefore, we have two possibilities.

Case 1: If $S(\pi_i) \leq S(\pi_j)$, we first consider the direction moving from state π_i to state π_j. From (31), we have $\alpha(\pi_i, \pi_j) = 1$. Thus

$$\theta(\pi_i)P(\pi_i, \pi_j) = Ce^{\frac{S(\pi_i)}{T_B}} q(\pi_i, \pi_j). \tag{A.1}$$

For the direction from state π_j to state π_j, we have

$$\alpha(\pi_j, \pi_i) = e^{\frac{S(\pi_i) - S(\pi_j)}{T_B}}. \tag{A.2}$$

Hence

$$\begin{aligned}\theta(\pi_j)P(\pi_j, \pi_i) &= Ce^{\frac{S(\pi_j)}{T_B}} q(\pi_j, \pi_i)\alpha(\pi_j, \pi_i)\\ &= Ce^{\frac{S(\pi_j)}{T_B}} q(\pi_j, \pi_i)e^{\frac{S(\pi_i) - S(\pi_j)}{T_B}}\\ &= Ce^{\frac{S(\pi_i)}{T_B}} q(\pi_j, \pi_i).\end{aligned} \tag{A.3}$$

Since $q(\pi_i, \pi_j) = q(\pi_j, \pi_i)$, therefore, from (A.1) and (A.3), we obtain the detailed balance equation.

Case 2: Now consider the scenario where $S(\pi_i) > S(\pi_j)$. Similarly, from (31), we have

$$\begin{aligned}\alpha(\pi_i, \pi_j) &= e^{\frac{S(\pi_j) - S(\pi_i)}{T_B}}\\ \alpha(\pi_j, \pi_i) &= 1.\end{aligned}$$

Thus

$$\begin{aligned}\theta(\pi_i)P(\pi_i,\pi_j) &= Ce^{\frac{S(\pi_i)}{T_B}}q(\pi_i,\pi_j)\\ &= Ce^{\frac{S(\pi_i)}{T_B}}q(\pi_j,\pi_i)e^{\frac{S(\pi_j)-S(\pi_i)}{T_B}}\\ &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i) \qquad \text{(A.4)}\\ \theta(\pi_j)P(\pi_j,\pi_i) &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i)\alpha(\pi_j,\pi_i)\\ &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i). \qquad \text{(A.5)}\end{aligned}$$

From (A.4) and (A.5), we have the detailed balance equation; therefore, the theorem follows. ■

REFERENCES

[1] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 6, pp. 1204–1216, Jul. 2000.
[2] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast with network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.
[3] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On delay performance gains from network coding," in *Proc. 40th Annu. Conf. Inf. Sci. Syst.*, 2006, pp. 864–870.
[4] T. Ho, M. Medard, M. Effros, and D. R. Karger, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Symp. Inf. Theory*, 2003, p. 442.
[5] S. Katti *et al.*, "XORs in the air: Practical wireless network coding," in *Proc. ACM SIGCOMM*, 2006, pp. 243–254.
[6] D. Nguyen, T. Nguyen, and X. Yang, "Wireless multimedia transmission with network coding," in *Proc. Packet Video Workshop*, 2007, pp. 326–335.
[7] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Res., Tech. Rep. MSR-TR-2004-78, 2004.
[8] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," in *Proc. IEEE Inf. Theory Workshop*, 2004, pp. 304–309.
[9] J. Widmer, C. Fragouli, and J.-Y. Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *Proc. Workshop Netw. Coding, Theory, Appl.*, 2005, pp. 1–6.
[10] S. R. Chandran and S. Lin, "Selective-repeat-ARQ schemes for broadcast links," *IEEE Trans. Commun.*, vol. 40, no. 1, pp. 12–19, Jan. 1992.
[11] J. L. Wang and J. A. Silvester, "Performance optimization of the go-back-N ARQ protocols over broadcast channels," *Comput. Commun.*, vol. 14, no. 7, pp. 393–404, Sep. 1991.
[12] T. Tran, T. Nguyen, and B. Bose, "A joint network-channel coding technique for single-hop wireless networks," in *Proc. 4th Workshop Netw. Coding, Theory, Appl.*, 2008, pp. 1–6.
[13] T. Tran, T. Nguyen, B. Bose, and V. Gopal, "A hybrid network coding technique for single-hop wireless networks," *IEEE J. Sel. Topics Adv. Commun.*, vol. 27, no. 5, pp. 685–698, Jun. 2009.
[14] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," in *Proc. Packet Video Workshop*, 2007, pp. 191–200.
[15] H. Seferoglu and A. Markopoulou, "Video-aware opportunistic network coding over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 713–728, Jun. 2009.
[16] J. Domzal, J. Dudek, P. Jurkiewicz, L. Romanski, and R. Wojcik, "The cross-protect router: Implementation tests and opportunities," *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 115–123, Sep. 2014.
[17] J. W. Roberts, "Internet traffic, QoS, and pricing," *Proc. IEEE*, vol. 92, no. 9, pp. 1389–1399, Sep. 2004.
[18] G. Miao and Z. Niu, "Bandwidth management for mixed unicast and multicast multimedia flows with perception based QoS differentiation," in *Proc. IEEE ICC*, 2006, pp. 687–692.
[19] T. Tran and T. Nguyen, "Adaptive network coding for wireless access networks," in *Proc. 19th ICCCN*, Aug. 2010, pp. 1–6.
[20] G. Perboli, R. Tadei, and L. Gobbato, "The multi-handler knapsack problem under uncertainty," Interuniversity Res. Cent., Montreal, QC, Canada, Rep. CIRRELT-2012-69, 2012.
[21] A. Sinclair, "Improved bounds for mixing rates of Markov chains and multicommodity flow," in *Combinatorics, Probability and Computing*. Berlin, Germany: Springer-Verlag, 1992.
[22] R. W. Yeung, S. Y. R. Li, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.
[23] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
[24] M. Medard, M. Effros, T. Ho, and D. Karger, "On coding for non-multicast networks," in *Proc. Allerton Conf. Commun.*, 2003, pp. 1–7.
[25] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, 2003, pp. 1–10.
[26] T. Ho *et al.*, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.
[27] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. 43rd Annu. Allerton Conf. Commun.*, 2005, pp. 1–10.
[28] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: An instant primer," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006.
[29] S. Deb *et al.*, "Network coding for wireless applications: A brief tutorial," in *Proc. IWWAN*, 2005, pp. 196–200.
[30] A. Douik, S. Sorour, M. Alouini, and T. Al-Naffouri, "Delay minimization for instant decodable network coding in persistent channels with feedback intermittence," *ArXiv e-Prints*, 2013.
[31] A. Douik, S. Sorour, M. Alouini, and T. Al-Naffouri, "Delay reduction in lossy intermittent feedback for generalized instantly decodeable network coding," in *Proc. IEEE Int. Conf. WiMob Comput., Netw. Commun.*, 2013, pp. 388–393.
[32] R. Koetter and M. Medard, "Beyond routing: An algebraic approach to network coding," in *Proc. IEEE INFOCOM*, 2002, pp. 122–130.
[33] Default TTL Values in TCP/IP, 2013. [Online]. Available: http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm
[34] IP Option Numbers [RFC791], [RFC1122], 2013. [Online]. Available: http://www.iana.org/assignments/ip-parameters/ip-parameters.xml
[35] I. Hou and P. R. Kumar, "Scheduling heterogeneous real-time traffic over fading wireless channels," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
[36] M. Mazzotti *et al.*, "Analysis of packet-level forward error correction for video transmission," in *Proc. IEEE VTC Spring*, 2011, pp. 1–5.
[37] C. Feng and B. Li, "Network coding for content distribution and multimedia streaming in P2P networks," in *Network Coding: Fundamentals and Applications*. San Diego, CA, USA: Academic, 2012.
[38] S. Sorour, A. Douik, S. Valaee, T. Y. Al-Naffouri, and M.-S. Alouini, "Partially blind instantly decodable network codes for lossy feedback environment," *IEEE Trans. Wireless Commun.*, vol. 13, no. 9, pp. 4871–4883, Sep. 2014.
[39] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Proc. IEEE Inf. Theory Appl. Workshop*, 2006, pp. 1–5.
[40] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[41] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, 2003, pp. 1–10.
[42] R. Aravind, M. Civanlar, and A. Reibman, "Packet loss resilience of MPEG-2 scalable video coding algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, pp. 426–435, Oct. 1996.
[43] R. Singh, A. Ortega, L. Perret, and W. Jiang, "Comparison of multiple description coding and layered coding based on network simulations," in *Proc. Dispersity Routing Store Forward Netw.*, 2000, pp. 929–939.
[44] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts: Process Scheduling*. Hoboken, NJ, USA: Wiley, 2010.
[45] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability benefit of network coding," Comput. Sci. Dept., Univ. Mass. Amherst, Amherst, MA, USA, Tech. Rep. 07-08, 2007.
[46] M. Jerrum and A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," in *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum ed. Boston, MA, USA: PWS, 1996.
[47] M. Abramowitz and I. A. Stegun, Eds., "Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables," in *The 9th Printing*. New York, NY, USA: Dover, 1972.
[48] Voice Over IP—Per Call Bandwidth Consumption. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html
[49] C. Wang, D. Koutsonikolas, Y. Hu, and N. Shroff, "FEC-based AP downlink transmission schemes for multiple flows: Combining the reliability and throughput enhancement of intra- and inter-flow coding," *Perform. Eval.*, vol. 68, no. 11, pp. 1118–1135, Nov. 2011.

Tuan Tran (M'XX) received the B.S. degree from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2000; the M.S. degree from the Polytechnic University of Turin, Turin, Italy, in 2006; and the Ph.D. degree from Oregon State University, Corvallis, OR, USA, in 2010, all in computer engineering.

He is currently an Assistant Professor with the College of Computer and Information Technology, Sullivan University, Louisville, KY, USA. Prior to that, he was a Postdoctoral Scholar with Arizona State University, Tempe, AZ, USA, and the University of Louisville from 2010 to 2012. His research interests include computer networks and cybersecurity, stochastic system modeling, network and channel coding, wireless systems, and multimedia networking.

Dr. Tran received the Best Paper Runner-Up Award at the IEEE International Conference on Computer Communication Networks in 2010 and the Jack Neubauer Memorial Award for the Best Systems Paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2012.

Thinh Nguyen (M'04) received the B.S. degree from the University of Washington, Seattle, WA, USA, in 1995 and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 2003.

He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA. He is interested in all things stochastic, with applications to signal processing, distributed systems, wireless networks, network coding, and quantum walks.

Dr. Nguyen has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the IEEE TRANSACTIONS ON MULTIMEDIA.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please check if “For the sake of expository” is properly changed to “For the sake of exposition”, to make the statement clear; otherwise, kindly provide the correction.
AQ2 = Please check if changes in this paragraph are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ3 = Please check if MC is properly introduced here; otherwise, kindly provide the correction.
AQ4 = Please check if “reflex” is properly changed to “reflect”; otherwise, kindly provide the correction.
AQ5 = Please check if changes in this sentence are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ6 = Please check if changes here are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ7 = Please check if citation for Fig. 15(a) should be captured as citation for Fig. 15 only, to match the figures; otherwise, kindly provide the correction.
AQ8 = Note that reference [3] and [7] are the same. Therefore, reference [7] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ9 = Note that reference [5] and [39] are the same. Therefore, reference [39] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ10 = Note that reference [26] and [38] are the same. Therefore, reference [38] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ11 = Please provide membership history of author “Tuan Tran.”
AQ12 = Kindly provide the area of study for both B.S. and Ph.D. degrees.

END OF ALL QUERIES

Context-Aware Interflow Network Coding and Scheduling in Wireless Networks

Tuan Tran, *Member, IEEE*, and Thinh Nguyen, *Member, IEEE*

***Abstract*—Recent approaches using network coding (NC) to mix data from different flows show significant throughput improvement in wireless networks. However, in this paper, we argue that exhaustively mixing packets from different flows may decrease network quality of service (QoS), particularly in the presence of flows with different service classes. We therefore propose a *context-aware interflow network coding and scheduling* (CARE) framework, which adaptively encodes data across the traffic to maximize the network QoS. First, we develop a perception-oriented QoS (PQoS) to measure the user satisfaction of different types of services. Next, based on the characteristics of the traffic, we optimally combine data across the flows and schedule the encoded packets in each time frame to maximize the PQoS at the receivers. Solving CARE is NP-hard; thus, we devise a computationally efficient approximation algorithm based on the Markov chain Monte Carlo method to approximate the optimal solution. We prove that the proposed approximation algorithm is guaranteed to converge to the optimal solution. The analytical and simulation results show that, under certain channel conditions, the proposed CARE-based schemes not only improve the network QoS but achieve high throughput across all receivers as well. Additionally, the results show that the approximation algorithm is efficient and robust to the number of data flows. In some transmission conditions, our CARE-based schemes can improve the network QoS up to 50% compared with the existing randomized NC techniques.**

***Index Terms*—Multiuser multiservice scheduling, quality of service (QoS), random network coding (RNC), wireless networks.**

I. Introduction

WE consider the problem of downlink transmission in wireless networks, whereby multiple data flows share a single wireless channel. Traditionally, data transmission is performed via the *store-and-forward* routing protocols in which an intermediate node stores incoming data and forwards them to the neighboring nodes toward the destinations without altering contents of the data. Differently, in the new *network coding* (NC) approach [1], an intermediate node is allowed to combine the incoming data packets before sending them out to the receivers. It has been shown that NC-based approaches significantly improve network performance, such as throughput [2]–[6], transmission delay [3], [7], and energy [8], [9]. For instance, Nguyen *et al.* [2], [6] showed that XOR-based NC schemes used in conjunction with a scheduler at a WiFi access point (AP) can significantly improve the network throughput of a broadcast session. The authors showed that, under some transmission conditions, bandwidth efficiency could be double compared with the traditional Auto Repeat reQuest (ARQ) approaches [10], [11]. Additionally, Eryilmaz *et al.* [3] have shown that, in unreliable wireless networks, transmission delay decreases substantially by using NC. Furthermore, Tran *et al.* [12], [13] showed that significant bandwidth gain can also be achieved by employing NC in conjunction with channel coding techniques across different unicast sessions. In a different avenue, Wu *et al.* [8] showed that NC can also be used to minimize the transmission energy in mobile ad-hoc networks as well.

In this paper, we focus on using NC-based approaches to improve the quality of service (QoS) of wireless networks that consist of multiple unicast flows. Generally speaking, providing high QoS for multiuser wireless networks is challenging. First, wireless links often suffer due to severe fading and interference. Additionally, channel conditions that usually change over time significantly affect the QoS at the receivers. Furthermore, the heterogeneity of channel conditions makes the scheduling problem more challenging, as receivers usually experience different data losses. Retransmitting lost data to a receiver in a bad channel condition may decrease the network bandwidth efficiency because data could be duplicated at other receivers. On the other hand, only serving receivers in good channel conditions could leave many other receivers in worse channel conditions with unacceptable QoS. The problem usually becomes combinatorially hard in nature.

In fact, there exists literature on using NC to improve network QoS. Notably, Seferoglu *et al.* [14], [15] proposed video-aware opportunistic XOR-based network encoding and scheduling schemes for video streaming. The proposed schemes take into account both video distortion and deadlines of the packets for optimal data encoding and scheduling. The simulation results showed significant video quality improvement [i.e., peak-signal-to-noise ratio (PSNR)] compared with the approaches without video-aware encoding. Those works, however, considered only the case where all data flows are multimedia streams (i.e., video). As a result, they may not work well when applied directly to scenarios where traffic includes both delay-sensitive (e.g., video streaming) and elastic applications (e.g., web browsing). In such a setting, the performance metric of

Manuscript received October 31, 2014; revised April 7, 2015, July 20, 2015, and November 17, 2015; accepted January 1, 2016. This work was supported in part by the National Science Foundation under Grant CNS-0845476 and Grant CNS-1547450 and in part by the Faculty Scholarship Grant of Sullivan University. This paper was presented in part at the 19th IEEE International Conference on Computer Communications and Networks (ICCCN), 2010. The review of this paper was coordinated by Prof. C. Assi.

T. Tran is with the College of Information and Computer Technology, Sullivan University, Louisville, KY 40205 USA (e-mail: ttran@sullivan.edu).

T. Nguyen is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA (e-mail: thinhq@eecs.oregonstate.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2016.2520361

delay-sensitive applications (e.g., PSNR for video streaming) might not be a good metric to measure the QoS of the delay-tolerant applications. Generally, the heterogeneity of the traffic content makes the encoding and scheduling problem more challenging because the transmitter needs to consider not only the transmission deadline but the *characteristics of the traffic* as well.

We consider the problem of multiuser downlink transmission of heterogeneous traffic in lossy wireless networks. A typical approach for such a system is to overprovision the QoS requirements for various flows. Such a system will work well as long as the aggregate demand from all the flows does not exceed the network capacity [16]. However, in an overloaded transmission scenario, the system performance is likely to deteriorate rapidly due to a traffic surge [17]. Therefore, we propose a new NC-based framework called Context-awARE Interflow Network Coding and Scheduling (CARE) which utilizes the user perception-oriented QoS model (PQoS) [18] for optimal interflow data encoding and scheduling. Generally speaking, CARE optimally encodes and schedules data transmission based on both channel conditions and characteristics of the traffic to maximize the network QoS from the user's perspective. Our results show that CARE significantly improves the network QoS and throughput, in comparison with the state-of-the-art NC-based approaches. The performance improvement basically comes from the optimal trade-off between the number of useful data packets transmitted by the deadline and the bandwidth allocation to different flows via its PQoS objective function. To the best of our knowledge, this is one of a few papers that consider PQoS as the objective function in formulating the multiuser downlink scheduling using NC in wireless networks.

Extending from our preliminary results in [19], the main contributions of this paper are summarized as follows.

- We first show that the approaches optimized for throughput might decrease the QoS at some receivers in the presence of traffic with different service classes. We then devise PQoS as the network metric to quantitatively measure the performance of different transmission strategies. The CARE framework based on PQoS consolidates not only the traffic priorities but also the characteristics of the flows in its encoding and scheduling operation. Analyses on QoS performance for different transmission strategies are provided in detail.
- We formulate CARE as a combinatoric optimization problem with constraints. We further show that CARE can be reduced to a set of weighted stochastic knapsack problems and, therefore, is NP-hard [20]. We then exploit the traffic characteristics to devise an efficient approximation algorithm based on the Markov chain Monte Carlo (MCMC) method for finding a near-optimal solution. Our results show that, under certain channel conditions and QoS requirements, our proposed CARE-based schemes lead to significant network performance improvement for both delay-sensitive and delay-tolerant data flows. We further provide analytical results on the asymptotic behavior and upper bound on the convergence time of the approximation algorithm based on the canonical path technique [21]. We also describe context-aware partial interflow NC (PCARE), an extension of CARE, which allows for finer grained bandwidth allocation.
- We provided theoretical analysis and intensive simulations to elaborate the system performance improvement of our proposed schemes. Our results reveal that optimizing the PQoS is equivalent to optimizing the network effective throughput constrained on the fairness condition among the users. The results also show that the approximation algorithm is efficient and robust to the number of data flows and quickly converges to the optimal solution.

The remainder of this paper is organized as follows. We first discuss some background and related work in Section II. In Section III, we describe the system model, the issues of the existing approaches, and performance metric. In Section IV, we analyze the system performance of different transmission strategies, CARE formulation, and its hardness. In Section V, we develop an approximation algorithm for CARE. Simulations and discussions are provided in Section VI. Finally, we conclude this paper in Section VII.

II. Background and Related Work

Random Network Coding: The notion of NC, i.e., mixing of data at intermediate nodes to increase the overall multicast throughput of a network, was first proposed in the seminal paper by Ahlswede *et al.* [1]. Its key idea is to prove the existence of some network codes (method of mixing data at intermediate nodes) that achieve multicast capacity. This spurted a number of works on construction of practical network codes, including algebraic, algorithmic, and randomized approaches [22]–[25]. In particular, the work in [22] proposed a class of linear network codes for multicast transmission. The author proved that linear coding suffices to achieve the maximum throughput, which is the max-flow min-cut from the source to each receiving node. Inspired by this work, Koetter and Medard [23] proposed a theoretical framework based on algebraic tools for deriving the conditions to achieve capacity in networks using linear codes. The proposed framework shows interesting connections between certain systems of polynomial equations and the solutions to network routing problems. Notably, the work in [4] (and its extended version in [26]) proposed a random NC (RNC) framework for efficient network code construction in a distributed manner. In particular, it shows that intermediate nodes in a network do not need to cooperate with each other to generate coded packets. Instead, each node independently generates its coded packets by combining its incoming data with the coding coefficients randomly selected from a large finite field. The authors proved that, with probability approaching 1, the encoded packets are independent. Based on this result, Chou *et al.* [25] proposed a practical solution to implement NC for an arbitrary network topology. In particular, it has been shown that, by inserting the coding coefficients into the coded packets' headers, the sinks can reconstruct the original data efficiently by solving a system of linear equations constructed by the encoded data.

Time slot	1		2		3		4
Scheme	UNI	RNC	UNI	RNC	UNI	RNC	UNI
Packet	a	c_1	b	c_2	a	c_3	b
D_1	x	x	-	o	o	o	-
D_2	-	o	x	x	-	o	o

Fig. 1. Example of UNI and RNC transmission schemes for two receiver scenarios. For the RNC scheme, coded packets $c_i = \alpha_i a + \beta_i b$. Packet receipts are denoted by "$\times$," "o," and "-" for lost, successful, and "received but useless" packets, respectively.

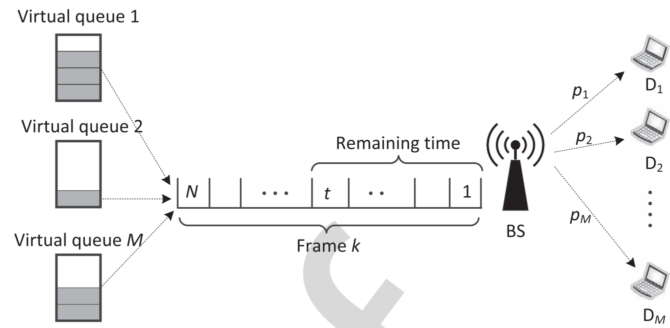


Fig. 2. System model. At the beginning of frame k, new arrived packets will be scheduled for transmission in the next N time slots (period of a frame).

RNC for Wireless Networks: It has been shown that applying NC in wireless ad-hoc networks can significantly improve the network bandwidth efficiency [27], transmission delay [28], [29], or transmission energy [7]. The key idea of these works is to exploit the nature of a broadcast signal, which can be intercepted by many neighboring nodes. Data of different flows are then combined (i.e., performing RNC) together to generate coded data packets before sending them to other nodes in the vicinity. Recently, Douik *et al.* [30], [31] have proposed techniques to reduce the decoding delay for different feedback constraints. These works, however, only focus on broadcasting transmission and do not consider the flows with heterogeneous content and services. A detailed list of wireless applications beneficial from using NC in these settings can be found in [29].

Our network model is closest to the model used in [2] and [6], whereby the authors model a broadcast session in a last-mile network. Differently, we consider multiple unicast flows sharing a bandwidth-constrained channel. Our transmission model can be applied to many practical transmission scenarios in WLAN, WiMAX, and cellular networks. Before describing our system model, we first illustrate the benefit of using NC in such a setting.

Example 1: An AP wishes to send two packets a and b to two receivers D_1 and D_2, respectively. In ARQ unicast protocol (UNI), the AP uses the first and second time slots to send a and b, respectively. As illustrated in Fig. 1, packet a is lost at D_1 while successfully received at D_2. However, D_2 discards a because it wants packet b instead. Similarly, in the second time slot, packet b is successfully received by D_1 (but it is discarded because D_1 wants a instead) while lost at D_2. Assuming that, in the next two time slots, the transmission links are in good conditions, then the AP can successfully retransmit the lost data packets to their intended receivers. As a result, it needs four time slots to deliver two packets a and b to D_1 and D_2, respectively.

Next, we consider a transmission scheme using RNC, as proposed in [32]. In this approach, the AP generates coded packets by linearly combining a and b with random coefficients and sending them out to the receivers. For example, coded packets c_i are generated as $c_i = \alpha_i a + \beta_i b$, $i = \{1, 2, 3, \ldots\}$, where α_i and β_i are coefficients drawn randomly from a large finite field F_q (q is the field size). The AP then broadcasts two coded packets c_1 and c_2 in the first and second time slots, respectively, as shown in Fig. 1. As a result, D_1 and D_2 will successfully receive c_2 and c_1, respectively. In the third time slot, the AP broadcasts another coded packet c_3, which is successfully received by both receivers. After three transmissions, each of the receivers now has two coded packets, which can be used to recover their desired data by solving a system of linear equations using the encoded packets. We note that the coefficients are included in the packets' headers, enabling the receivers to solve the linear equation system. It requires additional transmission overhead; however, with a sufficiently large packet size, this overhead is negligible [25]. Thus, the RNC approach needs only three transmissions to deliver two packets to the receivers, saving 25% transmission bandwidth compared with the UNI scheme.

III. System Model, Issues, and Performance Metric

A. System Model

We consider the problem of time-slotted downlink transmission of M data flows to M receivers (users) in lossy wireless networks, as illustrated in Fig. 2. We assume that the transmitter uses M "virtual queues" to store randomly arrived packets of different flows before sending them out to the corresponding receivers. Transmissions are scheduled by frames (periods), each consisting of N time slots. Generally, the value of N can be determined by the hard deadline of buffered data or by the number of backlogged data packets in the buffer [15], [33]–[35]. When a new flow arrives at the transmitter in the current transmission frame, it will be scheduled for transmission in the next frame. Without loss of generality, we assume that there are $k_i, i = 1, 2, \ldots, M$ data packets being delivered to receiver D_i in each transmission frame. In this paper, we focus on finding an optimal flow partition scheme for encoding and scheduling data across different flows, given a set of data packets for each transmission frame.

We assume that the scheduled packets of delay-sensitive applications, which cannot be delivered to the corresponding receivers by the deadline, will be discarded from the system. The system QoS is computed based only on the number of data packets that have been received successfully within that period. On the other hand, for delay-tolerant reliable applications such as file transfer, the lost packets will be retransmitted until they are successfully delivered (possibly via multiple transmission frames). Detail of the mathematical formula used to compute the system QoS is defined in Section III-C3. In addition, we assume that the transmission links between the transmitter and the receivers are heterogeneous and independent identically distributed binary erasure channels with erasure probability p_i. A flow i implements a packet-level forward error-correcting (FEC) code (n_i, k_i) (e.g., [36]) to cope with data errors. Using such an FEC code, receiving k_i out of n_i transmitted packets is sufficient for receiver D_i to recover the original information. The code rates k_i/n_i are prespecified based on the network conditions or/and priority of the users' subscription. Finding the optimal coding rates is beyond the scope of this paper. Additionally, we assume that the transmitter has enough memory to store data for at least one transmission frame $N = \sum_{i=1}^{M} n_i$.

Furthermore, in our model, each receiver maintains a *decoding buffer* for storing coded packets received within a frame. By the end of each data frame, a receiver will decode the received packets and push them to the upper Open Systems Interconnection (OSI) layer above for further processing. It is important to note that the use of a decoding buffer at receivers is a standard assumption, which has been adopted for several years in NC-based techniques and literature (e.g., [5], [6], [15], [26], [30], [37], [38], and references therein).

B. What Could Go Wrong With Interflow Network Coding?

As observed in *Example 1*, mixing packets from different flows is clearly beneficial. However, *should one advocate mixing at every opportunity*? The answer should be "No." In particular, Wu *et al.* were the first to consider this mixing problem in a different context [39]. Intuitively, exhaustively mixing the data of several flows may decrease the chance that a receiver can recover its information. To illustrate this point, we show a simple counterexample as follows.

Counterexample: We consider the same setup, as shown in Fig. 1. In addition, we assume that packet losses at D_1 and D_2 are independent and follow the Bernoulli trial with $p_1 = 1/3$ and $p_2 = 2/3$, respectively. To transmit data reliably, it makes sense for the transmitter to employ stronger protection for the data intended to D_2. Thus, suppose that two packet-level FEC codes $(n_1, k_1) = (3, 2)$ and $(n_2, k_2) = (3, 1)$ are used for the flows to D_1 and D_2, respectively.[1] We recall that, by using a code (n, k), the transmitter uses n time slots to transmit k information packets ($n \geq k$), whereby receiving any k packets out of the n transmitted packets is sufficient to recover the k original packets. Suppose that the transmitter has $n_1 + n_2 = 6$ time slots for delivering three packets, i.e., two for D_1 and one for D_2. In a non-mixing technique, i.e., packets from different flows are sent separately, the probability that all the receivers recover their desired data is computed as

$$P_{\mathrm{UNI}} = \prod_{i=1}^{2} \sum_{j=0}^{r_i} \binom{n_i}{j} p_i^j (1 - p_i)^{n_i - j} \tag{1}$$

where $r_i = n_i - k_i$ for $i = \{1, 2\}$. Substituting values of p_i, n_i, and k_i into (1), we have $P_{\mathrm{UNI}} = 0.5213$, which is about 1/2 packet per time slot.

On the other hand, in an RNC-based technique, all packets are mixed to produce coded packets. In such a transmission strategy, each receiver needs to receive at least three coded packets correctly, to recover its desired information [41]. The transmitter has six time slots for delivering the *coded* packets to both receivers. The probability that both receivers recover their desired data is given by

$$P_{\mathrm{RNC}} = \prod_{i=1}^{2} \sum_{j=0}^{r} \binom{N}{j} p_i^j (1 - p_i)^{N - j} \tag{2}$$

where $r = \sum_{i=1}^{2} (n_i - k_i) = 3$ and $N = \sum_{i=1}^{2} n_i = 6$. Substituting the values of p_i for $i = \{1, 2\}$ into (2), we obtain $P_{\mathrm{RNC}} = 0.2876$. This is approximately equivalent to a throughput of 1/5 packet per time slot, which is about 2.5 times less than that of the non-NC technique earlier. Obviously, applying NC in this case decreases the network throughput.

C. Perception-Oriented QoS

We next describe how the PQoS metric is constructed to measure the performance of different transmission strategies. Roughly speaking, PQoS function is used to estimate the service satisfaction at each user. Thus, PQoS depends not only on the number of received packets within a time period but also on the type of service (ToS). For the sake of exposition, we categorize the network traffic into two types: delay-sensitive traffic (e.g., video streaming, audio streaming, etc.) and elastic traffic (e.g., file transfer, e-mail, etc.).[2] We note, however, that one can easily extend the framework to the cases of more than two types of traffic, albeit a more sophisticated model.

1) PQoS for Delay-Sensitive Traffic: In our model, delay-sensitive traffic (e.g., video streaming) is encoded into multiple layers, e.g., a base layer and enhancement layers [42], [43]. In such a model, the base layer must be presented to present other enhancement layers. Thus, to maintain a minimal QoS, a receiver needs to receive at least N_0 packets of the base layer per transmission frame. When the number of received packets is fewer than N_0, the QoS at the receiver decreases significantly. The reason is that, in such an encoding scheme, the QoS at the receiver is mainly contributed by the base layer. On the other hand, when more packets of the enhancement layers are received, the QoS at the receiver only increases slightly due to only additional details of the information added into the base frame. We extend the satisfaction function proposed in [18] to model the PQoS for delay-sensitive applications. Mathematically, we can represent it in (3), shown at the bottom of the next page, where S_i denotes the number of packets received successfully, and N_0 denotes the minimum number of packets to maintain a satisfaction factor of $\gamma_0 \in [0, 1]$. In this formulation, the first case indicates that, when the number of received packets is fewer than the threshold $\gamma_0 N_0$, the value of PQoS is equal to zero. The intuition can be reasoned in the context of layered video transmission (e.g., MPEG-2) where the base layer is unrecoverable. Thus, no information is displayed,

[1]We note that the packet-level FEC codes here can be viewed as the raptor codes [40] with n_i and k_i, respectively, being the output and original symbols.

[2]We use the terms "elastic" and "delay-tolerant" interchangeably.

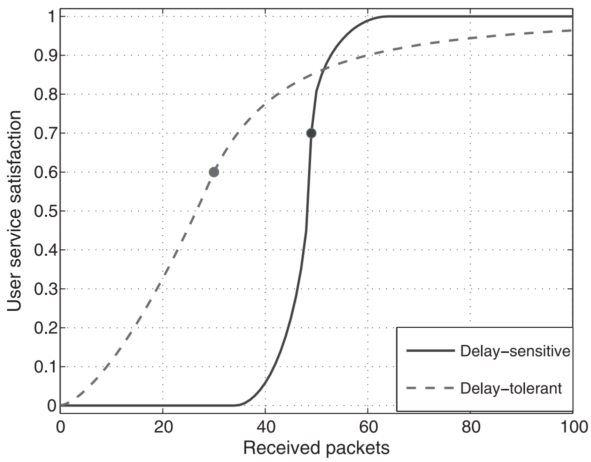


Fig. 3. Satisfaction functions of different applications versus the number of received packets.

resulting in a zero PQoS. The second case accounts for the scenario when the number of received packets is greater than the minimum threshold $\gamma_0 N_0$ but less than N_0. In this case, the PQoS steeply decreases due to only a part of the base layer recovered. The third case indicates that the value of PQoS increases significantly when the number of received packets is greater than N_0. In this case, more detail of the video frame is added to achieve high-definition display. Finally, the last case accounts for a scenario wherein the missing data add negligible QoS to the data flow, resulting in a high value of PQoS close to one. An example of the PQoS function for delay-sensitive application, with respect to the number of received packets, is illustrated by the blue solid curve in Fig. 3.

2) PQoS for Delay-Tolerant Traffic: We use different functions to model elastic traffic, as illustrated by the red dashed curve in Fig. 3. For such a flow, the parameter N_0, i.e., the minimum data received per frame, is viewed as the minimum bandwidth allocated to that flow. When the number of received packets is fewer or greater than N_0, respectively, the satisfaction function decreases or increases slightly. We note that, in delay-tolerant traffic, if a transmitted packet is lost during transmission, a negative acknowledgement (NACK) will be sent from the receiver to the transmitter for retransmission (possibly in other frames). Thus, for reliable data flows (e.g., data file downloading), every byte of the information will be reliably delivered to the receiver. Mathematically, the PQoS for elastic traffic is given as

$$\gamma_i = \mathcal{F}(S_i) = \begin{cases} \gamma_0 . \left(\frac{S_i}{N_0}\right)^2, & S_i < N_0 \\ 1 - (1-\gamma_0)\left(\frac{N_0}{S_i}\right)^2, & S_i \geq N_0. \end{cases} \quad (4)$$

In the first case, when the number of received packets is fewer than N_0, the value of PQoS decreases slightly by a factor of the ratio of S_i and N_0. It is worth noting that this ratio is less than one; thus, its square would result in a smaller value. On the other hand, when $S_i \geq N_0$, PQoS increases slightly. The intuition of the proposed PQoS functions is to reflect the fact that a slightly longer or shorter delay in receiving a delay-tolerant data packet, e.g., e-mail, does not affect much to the user's perception of the QoS.

3) Performance Metric: We use the average network PQoS as our metric to compare the performance of different transmission strategies. The average network PQoS is computed by averaging the expected PQoS across all the users for each transmission frame over infinite system runtime. Mathematically, we have that

$$\gamma = \lim_{\tau\to\infty} \frac{1}{\tau M} \sum_{\tau} \sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \quad (5)$$

where τ is the operation time of the network divided into several transmission frames, M is the number of data flows, c_i is the flow priority, and $\mathbb{E}[\gamma_i]$ denotes the expected PQoS of user i in one transmission frame. The $\lim_{\tau\to\infty}$ indicates that the expectation of the system performance is computed over a long time interval. Given the satisfaction functions, we call a transmission technique the best, if it has the largest expected PQoS across all users.

IV. Performance Analysis

In this paper, we analyze different transmission strategies, depending on how the transmitter exploits the characteristics of the traffic. In particular, we analyze the performance of the traditional UNI, systematic RNC (SRNC), and CARE.

A. Unicast

In this technique, data packets of different information flows are transmitted separately in a round-robin fashion [44]. In each period (frame), the transmitter allocates n_i time slots to transmit k_i data packets to receiver D_i [i.e., packet-level FEC code (n_i, k_i)]. If there is a packet loss, the transmitter uses the $n_i - k_i$ redundant time slots to retransmit the lost packets. The transmitter switches to transmit data packets for other users, when it receives an acknowledgment (ACK) message from D_i indicating that all data have been received successfully, or all time slots allocated for it have been used. Considering flow i destined to receiver D_i and letting p_i denote the packet erasure

$$\gamma_i = \mathcal{F}(S_i) = \begin{cases} 0, & S_i < \gamma_0 N_0 \\ \gamma_0\left(1 - \frac{\sqrt{(N_0 - S_i)(N_0 + S_i - 2\gamma_0 N_0)}}{(1-\gamma_0)N_0}\right), & \gamma_0 N_0 \leq S_i < N_0 \\ \gamma_0 + (1-\gamma_0)\frac{\sqrt{(S_i - N_0)[N_0 - S_i + 2N_0(1-\gamma_0)]}}{(1-\gamma_0)N_0}, & N_0 \leq S_i < (2-\gamma_0)N_0 \\ 1, & S_i \geq (2-\gamma_0)N_0 \end{cases} \quad (3)$$

probability of the transmission link, the probability that receiver D_i recovers all its data can be written as

$$P_i^s = \sum_{j=k_i}^{n_i} \binom{n_i}{j} p_i^{n_i-j}(1-p_i)^j \tag{6}$$

where $\binom{n_i}{j}$ denotes the number of combinations of size j out of n_i elements. We further note that, by using UNI, i.e., data transmitted separately, receiver D_i could recover a fraction of the original data, if the number of received packets is fewer than k_i. Let the random variables X and Y, respectively, be the number of original packets and the total number of received packets. The probability that D_i obtains only m original packets and the total received packets is fewer than k_i is given as

$$\begin{aligned} P_i(m) &= P(X=m, Y<k_i) \\ &\overset{(a)}{=} \sum_{l=m}^{k_i-1} \Pr(Y=l|X=m)\Pr(X=m). \end{aligned} \tag{7}$$

In this case, Y is equal to X plus the number of coded packets received during the second stage transmission while the sum runs over all the possible values of Y. We note that step (a) consists of two parts: 1) the conditional probability of receiving $l = m, \ldots, k_i - 1$ data packets in total, given m original data packets received; 2) the probability that m of k_i original data packets are received. In the extreme case, i.e., $l = m$, it implies that none of the coded packets is received. On the other hand, the maximum number of coded packets is $k_i - 1 - m$, corresponding to the case of $k_i - 1$ data packets received in total. Furthermore, the probability that only m of k_i original data packets are received is written as

$$\Pr(X=m) = \binom{k_i}{m}(1-p_i)^m p_i^{k_i-m}. \tag{8}$$

Given that only m original data packets are received, the probability that only $l < k_i$ data packets are received over n_i transmissions is given by

$$\Pr(Y=l|X=m) = \binom{n_i-k_i}{l-m}(1-p_i)^{l-m} p_i^{n_i-k_i-(l-m)}. \tag{9}$$

Combining (8) and (9), we have that

$$\begin{aligned} &P(Y=l|X=m)P(X=m) \\ &= \binom{k_i}{m}(1-p_i)^m p_i^{k_i-m}\binom{n_i-k_i}{l-m} \\ &\quad \times (1-p_i)^{l-m} p_i^{n_i-k_i-(l-m)} \\ &= \binom{k_i}{m}\binom{n_i-k_i}{l-m}(1-p_i)^l p_i^{n_i-l}. \end{aligned} \tag{10}$$

Therefore, the probability that the m original data packets and the total $l < k_i$ data packets are received can be computed as

$$P_i(m) = \sum_{l=m}^{k_i-1} \binom{n_i-k_i}{l-m}\binom{k_i}{m} p_i^{n_i-l}(1-p_i)^l. \tag{11}$$

Let γ_i denote the PQoS value of receiver D_i; therefore, from (6) and (7), the expected PQoS across all the users can be written as

$$\begin{aligned} \gamma &= \frac{1}{M}\mathbb{E}\left[\sum_{i=1}^{M} c_i\gamma_i\right] = \frac{1}{M}\sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \\ &= \frac{1}{M}\sum_{i=1}^{M} c_i\left(\mathcal{F}(k_i)P_i^s + \sum_{m=0}^{k_i-1}\mathcal{F}(m)P_i(m)\right) \end{aligned} \tag{12}$$

where $c_i \in (0, 1]$ denotes the weighted factor (priority) for the ith flow (this factor can be justified via the cost that the user D_i pays to the service provider); $\mathbb{E}[.]$ denotes the expected function; and $\mathcal{F}(\cdot)$ is defined in (3) and (4), depending on the type of the data flow.

B. Systematic Random Network Coding

Transmission in SRNC is classified into base and augmentation phases. In the base phase, all $\sum_{i=1}^{M} k_i$ original packets will be transmitted. The receivers cache all the received packets, including packets that are not intended to them. By keeping data packets intended to others, a receiver can use them to decode its desired data in the second phase. Next, in the augmentation phase, the transmitter combines the data packets of all flows to generate coded packets and broadcasts them to the receivers over $N - \sum_{i=1}^{M} k_i$ redundant time slots. The intuition of using systematic coding that transmits data in two phases is that the receivers that lose some original packets can receive more coded packets to decode their own data using the RNC method [41]. On the contrary, receivers that are unable to obtain a full set of data packets can still recover partial data from the original packets transmitted in the base phase. Such a transmission has been discussed in [45], and generally, it will result in higher performance compared to the RNC.

In SRNC, a receiver D_i can recover k_i desired packets, if it correctly receives either all k_i original packets or a full set of $K = \sum_{i=1}^{M} k_i$ packets (either original or coded packets). Let P_i^s denote the probability that D_i can recover all its data, and let the random variables U and V denote the original and total received packets at receiver D_i, respectively. We have that

$$\begin{aligned} P_i^s &= P(U=k_i, V<K) + P(U\le k_i, V\ge K) \\ &= (1-p_i)^{k_i}\left[\sum_{l=0}^{K-k_i-1}\binom{N-k_i}{l}p_i^{N-k_i-l}(1-p_i)^l\right] \\ &\quad + \sum_{j=0}^{k_i}\binom{k_i}{j}p_i^{k_i-j}(1-p_i)^j \\ &\quad \times \sum_{t=K-j}^{N-k_i}\binom{N-k_i}{t}p_i^{N-k_i-t}(1-p_i)^t. \end{aligned} \tag{13}$$

In (13), the first term accounts for the case when all the k_i original data packets are received successfully during the base phase transmission. The second term expresses the probability that j original packets are received during the base phase and

t coded packets are received during the augmentation phase, where $t = K - j, \ldots, N - k_i$. In this case, at least K data packets (both original and coded packets) are received by D_i; thus, it can recover its data by solving the system of linear equations formed by the received data packets.

We further note that D_i may receive only partial of the original data during the base phase. We denote $P_i(m)$ being the probability that D_i receives m original packets ($m < k_i$), and its total number of packets received correctly is less than K. Similar to (7), the probability of partial data recovery of D_i is written by

$$\begin{aligned} P_i(m) &= P(U = m, V < K) \\ &= \sum_{l=m}^{K-1} \binom{N-k_i}{l-m} \binom{k_i}{m} p_i^{N-l}(1-p_i)^l. \end{aligned} \tag{14}$$

In such cases, D_i cannot recover all its data, and only data received during the base phase contribute to the QoS of flow i. From (13) and (14), we express the expected PQoS across all receivers as follows:

$$\begin{aligned} \gamma &= \frac{1}{M}\mathbb{E}\left[\sum_{i=1}^{M} c_i\gamma_i\right] = \frac{1}{M}\sum_{i=1}^{M} c_i \mathbb{E}[\gamma_i] \\ &= \frac{1}{M}\sum_{i=1}^{M} c_i \left(\mathcal{F}(k_i)P_i^s + \sum_{m=0}^{k_i-1}\mathcal{F}(m)P_i(m)\right). \end{aligned} \tag{15}$$

C. Proposed Context-Aware Interflow Network Coding and Scheduling

1) Main Idea: Instead of combining all flows together, the transmitter now selectively chooses the flows to be mixed based on the channel conditions, ToS, and priorities of the flows. We assume that the M incoming data flows are partitioned into G groups; then, for each group, the transmitter uses SRNC to transmit data to the receivers within that group. The objective of the transmitter is to determine the optimal partition (i.e., which flows are combined together) to maximize the PQoS across all users. It is clear that mixing packets from all incoming flows could decrease the system performance due to *mismatch in ToS, priorities, and channel conditions*. On the other hand, mixing packets of flows with similar characteristics could increase the network performance. A precise mathematical formulation of CARE will be described as follows.

2) CARE Formulation: Let $\mathcal{G}$ denote a partition of the incoming information flows and $|\mathcal{G}|$ denote the number of groups in $\mathcal{G}$. Let M_i denote the number of data flows in group $i, i = 1, \ldots, |\mathcal{G}|$. Per each group, we use the SRNC technique described earlier to transmit the data. Consider the ith group, and let $N_i = \sum_{j=0}^{M_i} n_{ij}$ and $K_i = \sum_{j=0}^{M_i} k_{ij}$, respectively, denote the total number of available time slots and information packets being transmitted for group i. Here, (n_{ij}, k_{ij}) denotes the packet-level FEC of flow delivered to receiver j of group i. We note that flow j of group i, i.e., f_{ij}, is nothing but just some original data flow f_t, where the index has been relabeled. Thus, the FEC code (n_{ij}, k_{ij}) is also just a relabeled version of the original FEC code (n_t, k_t). Similarly, as computed in SRNC technique, the probability that receiver j of group i, i.e., D_{ij}, can recover its desired data is given as

$$\begin{aligned} P_{ij}^s = (1-p_{ij})^{k_{ij}} &\left[\sum_{l=0}^{K_i-k_{ij}-1} \binom{N_i-k_{ij}}{l} p_{ij}^{N_i-k_{ij}-l}(1-p_{ij})^l\right] \\ &+ \sum_{s=0}^{k_{ij}} \binom{k_{ij}}{s} p_{ij}^{k_{ij}-s}(1-p_{ij})^s \\ &\times \sum_{t=K_i-s}^{N_i-k_{ij}} \binom{N_i-k_{ij}}{t} p_{ij}^{N_i-k_{ij}-t}(1-p_{ij})^t. \end{aligned} \tag{16}$$

In this equation, the first term accounts for the case where original packets are successfully received during the basis transmission phase of group i, whereas the second term expresses the probability that at least K_i data packets (including both original and coded packets) are received successfully after both phases of transmission.

Similarly, it is straightforward to calculate the probability that receiver D_{ij} recovers m out of k_{ij} original packets ($m < k_{ij}$). That is

$$P_{ij}(m) = \sum_{l=m}^{K_i-1} \binom{N_i-k_{ij}}{l-m} \binom{k_{ij}}{m} p_{ij}^{N_i-l}(1-p_{ij})^l. \tag{17}$$

Let a random variable γ_{ij} denote the PQoS of receiver D_{ij}. Then, the expected PQoS over all users is given by

$$\gamma(\mathcal{G}) = \frac{1}{M}\mathbb{E}\left[\sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i} c_{ij}\gamma_{ij}\right]. \tag{18}$$

Therefore, a partition scheme is optimal, if it maximizes the expected PQoS across all users. The CARE optimization problem can be formulated as

$$\begin{aligned} \text{CARE}: \quad & \max_{\mathcal{G}\in\Omega} \left\{\frac{1}{M}\sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i} c_{ij}\mathbb{E}[\gamma_{ij}]\right\} \\ \text{s.t.}: \quad & \sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i} n_{ij} = N \qquad (19) \\ & \sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i} k_{ij} = K \qquad (20) \\ & 0 \le c_{ij} \le 1 \text{ for } i = 1, 2, \ldots, |\mathcal{G}| \\ & j = 1, 2, \ldots, M_i \qquad (21) \\ & \mathbb{E}[\gamma_{ij}] = \mathcal{F}(k_{ij})P_{ij}^s + \sum_{m=0}^{k_{ij}-1}\mathcal{F}(m)P_{ij}(m) \qquad (22) \end{aligned}$$

where Ω denotes the collection of all nonempty-subset partitions of M flows. The objective is to maximize the average expected PQoS across all the receivers. The first constraint represents the maximum number of time slots available for transmission in a data frame. The quantity $\sum_{j=1}^{M_i} n_{ij}$ gives the number of time slots allocated for group $i, i = 1, \ldots, |\mathcal{G}|$. The second constraint accounts for the total number of original data packets that need to be transmitted, i.e., $\sum_{j=1}^{M_i} k_{ij}$. Finally, the last two constraints, respectively, express the weight factor and the equation to compute the expected PQoS based on the type of applications and the number of received packets for receiver j in group i.

3) CARE Hardness: We next show that finding the optimal solution to the CARE problem is NP-hard. In particular, we will show 1) an existing algorithm that reduces an instance of CARE to an instance of the stochastic reward knapsack problem (SKP) [20] in polynomial time, and show that, 2) given the solution of CARE, the solution of the SKP can be found in polynomial time.

SKP Description: The SKP problem is described as follows. Given a set of n items, where item i has a fixed weight w_i and a random value v_i, the distribution of v_i could be unknown. The objective is to select a subset of items such that it maximizes the sum values while not exceeding the limit capacity of the knapsack.

Instance Reduction: We assume that there are M information flows, represented by their FEC codes $(n_i, k_i), i = 1, \ldots, M$, that we want to deliver to M receivers within N time slots. Without loss of generality, we assume that $\sum_{i=1}^{M} n_i > N$, i.e., only a subset of the M flows is selected. The expected PQoS of each subgroup of the M flows is considered as reward v_i of an item of the SKP problem. It is clear that $v_i \in V$ is a random variable, where its value depends on the erasure probability p_i and the flow partition. For simplicity, we let the priority factor $c_i = 1 \; \forall\, i = 1, \ldots, M$. Therefore, we have an instance of the SKP corresponding to a partition of the M flows into subgroups.

Solution Reduction: Finding the solution of SKP, given the solution of CARE, is straightforward. Let us assume that $\mathcal{G}^*$ is a partition that maximizes the expected PQoS across all the users. We have that

$$\begin{aligned}\sum_{i=1}^{|\mathcal{G}^*|}\sum_{j=1}^{M_i}\mathbb{E}[\gamma_{ij}] &\overset{(a)}{\geq} \sum_{i=1}^{|\mathcal{G}|}\sum_{j=1}^{M_i}\mathbb{E}[\gamma_{ij}] \quad \forall\, \mathcal{G} \in \Omega \\ &\overset{(b)}{\geq} \sum_{i=1}^{M} x_i v_i \quad \forall\, v_i \in V \\ x_i = \{0,1\}, &\sum_{i=1}^{M} x_i w_i \leq N. \end{aligned} \tag{23}$$

The inequality (a) follows from the assumption that $\mathcal{G}^*$ is the solution to CARE, i.e., the optimal one that maximizes the expected PQoS. The inequality (b) immediately holds because the right-hand side of (b) is equivalent to that of (a), representing in the context of SKP, i.e., the value distribution of the items. Thus, the union of the subsets of $\mathcal{G}^*$ forms the subsets of items selected for the SKP, where the total value that is maximized with the weight is less than the knapsack capacity.

D. Context-Aware Partial Interflow Network Coding

We next discuss PCARE, an extension of CARE, to further improve the system PQoS. We want to emphasize that PCARE has different formulation in comparison with CARE. On one hand, CARE seeks for optimal mixing and scheduling of incoming data flows, whereby a flow (as a whole) can be either combined with other flows or transmitted separately. On the other hand, PCARE allows incoming data flows to be further divided into subflows, which consequentially are combined with other subflows (of other flows) for transmission. By doing so, PCARE can achieve a finer grained bandwidth allocation compared with CARE, albeit a more sophisticated computation. Fig. 4 illustrates an example of subflow partitioning and mixing of PCARE. In this example, we consider three incoming data flows with their corresponding FEC codes being $(n_1, k_1) = (3, 2)$, $(n_2, k_2) = (1, 1)$, and $(n_3, k_3) = (4, 3)$, respectively. We recall that an FEC code (n, k) implies that n time slots are used to deliver k data packets, where receiving any k out of n transmitted packets is sufficient to recover the data. The PCARE scheme is performed via the following steps.

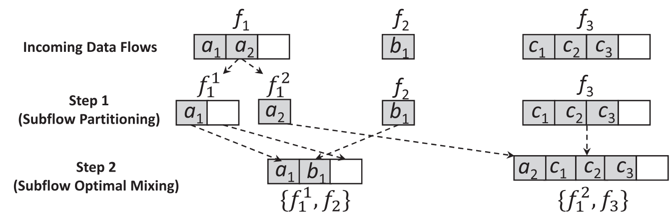


Fig. 4. Example of subflow partitioning and mixing of PCARE.

- *Subflow Partitioning*: PCARE first divides incoming data flows into subflows. We emphasize that there are many ways to divide a flow $f_i : (n_i, k_i)$ into subflows. Let $\mathbf{f}$ be the set representing all subflows of f_i. Consider a configuration where f_i is divided into J subflows $f_i^j, j = 1, \ldots, J$. We have the following constraints:

$$\mathbf{f} \supset f_i^j : \left(n_i^j, k_i^j\right), j = 1, \ldots, J$$
$$n_i = \sum_{j=1}^{J} n_i^j, k_i = \sum_{j=1}^{J} k_i^j.$$

  In Fig. 4, flow f_1 is divided into $J = 2$ subflows $f_1^1 : (n_1^1, k_1^1) = (2, 1)$ and $f_1^2 : (n_1^2, k_1^2) = (1, 1)$, while we keep flows f_2 and f_3 intact (a flow is a subflow of itself). As shown, even with this simple example, there are many ways to divide the incoming flows into subflows. Our example in Fig. 4 shows only one specific subflow configuration.
- *Subflow Optimal Mixing*: Next, given a set of subflows, they are then optimally mixed together for transmission. In our example in Fig. 4, subflow f_1^1 is combined with flow f_2, whereas subflow f_1^2 is combined with flow f_3. Data packets of each subgroup are then combined together within that subgroup for transmission. The key idea of PCARE is to divide the incoming flows into subflows, enabling finer grained bandwidth allocation for each subgroup. To illustrate this point, we next show a simple numerical example to illustrate the benefit of subflow partitioning of the PCARE scheme.

Numerical Example: Assume that $p_1 = 0.06$, $p_2 = 0.2$, and $p_3 = 0.05$ are the packet loss rates of the channels to receivers

D_1, D_2, and D_3, respectively. Supposing that we use RNC for combining data within each subgroup, we then have the probability that all receivers can recover their desired data using the PCARE scheme in Fig. 4, which is computed as follows.

- *Receiver D_1*: The probability that receiver D_1 can recover its desired data is computed by the product of the probabilities that data in each subgroup can be recovered successfully. In our example in Fig. 4, there are two subgroups, which have two and four data packets being transmitted in three and five time slots, respectively. The probability that D_1 can recover its data is computed as

$$P_1 = \sum_{i=2}^{3} \binom{3}{i} (1-p_1)^i p_1^{3-i} \times \sum_{i=4}^{5} \binom{5}{i} (1-p_1)^i p_1^{5-i}. \quad (24)$$

  The first and second terms represent the probability that D_1 successfully receives data packets a_1 and a_2 in the first and second subgroups, respectively.
- *Receivers D_2 and D_3*: Similarly, we can compute the probabilities that receivers D_2 and D_3 can recover their desired data as

$$P_2 = \sum_{i=2}^{3} \binom{3}{i} (1-p_2)^i p_2^{3-i} \quad (25)$$

$$P_3 = \sum_{i=4}^{5} \binom{5}{i} (1-p_3)^i p_3^{5-i}. \quad (26)$$

Substituting the values of $p_1 = 0.06$, $p_2 = 0.2$, and $p_3 = 0.05$ to (24)–(26), we obtain $P_1 = 0.9581$, $P_2 = 0.8960$, and $P_3 = 0.9974$. As a result, the probability that all receivers can recover their desired data is $P = P_1 \times P_2 \times P_3 = 0.8391$.

On the other hand, the best solution CARE can be achieved by combining flows 1 and 3 together, while transmitting flow 2 separately. Using this scheme, all receivers can recover their desired data with a probability $P = 0.792$, which is much lower than that of the PCARE scheme earlier. Therefore, we can see that, by dividing traffic flows into subflows, PCARE achieves better bandwidth allocation, resulting in higher system performance. However, one can also see that finding the optimal solution to the PCARE scheme is extremely computationally expensive. Its detailed theoretical analysis is considered as our future investigation.

V. Approximation Algorithm to Context-Aware Interflow Network Coding and Scheduling

Here, we describe a simple but efficient heuristic algorithm based on the well-known MCMC method [46] to find a near-optimal solution. Although MCMC-based techniques have been extensively used to solve hard optimization problems, it is very challenging to devise an efficient algorithm to approximate the solution of a given specific problem. Typically, a system designer needs to 1) design the target distribution that reflects the solution and 2) effectively generate samples from this target distribution. Unfortunately, achieving such design goals is challenging because it is very difficult to know the stochastic properties of the system. Additionally, it is not trivial to generate samples from an arbitrary distribution and design mechanism to transition among the states. Furthermore, the convergence time of the proposed algorithm needs to be upper bounded to ensure the performance guarantee. Here, we will describe how to obtain such objectives. We start with the description of the MCMC-based approximation algorithm and then prove its upper bounded convergence time.

A. MCMC-Based Algorithm (CARE-SAB)

Here, we show how to appropriately construct a target distribution and use MCMC to obtain the solution. Consider a scenario with M concurrent flows traversing through the base station. Let Ω be the set of all possible partition policies and $S(\pi)$ be the average satisfaction factor of a partition policy $\pi \in \Omega$. We represent each partition policy by an M-tuple group index as $\pi = (i, j, \ldots, k)$, where i indicates that the first flow belongs to group i, the second flow belongs to group j, and so on. The objective is to maximize the average PQoS over all users. That is

$$\max_{\pi \in \Omega} S(\pi) = \max_{\pi \in \Omega} \left\{ \sum_{i=1}^{|\pi|} \sum_{j=1}^{M_i} c_{ij} \gamma_{ij} \right\}. \quad (27)$$

We should note that the size of Ω, i.e., the number of ways that M flows can be partitioned into subgroups, is very large. Based on the result of [47], we have that

$$|\Omega| = \sum_{j=1}^{M} \frac{1}{j!} \sum_{i=0}^{j} (-1)^i \binom{j}{i} (j-i)^M. \quad (28)$$

Hence, using exhaustive search, even for a reasonably small number of flows, is infeasible for time-sensitive applications. Moreover, every time a flow joins, terminates, or its channel condition changes, the AP needs to repartition again. Instead, by using the MCMC method, we will show that the time to achieve the near-optimal solution will be substantially reduced.

We first define the target distribution as follows:

$$f(\pi) = C e^{\frac{S(\pi)}{T_B}} \quad (29)$$

where C is a normalization factor, and T_B is a "cooling" parameter that controls the process convergence. In particular, when T_B reaches to a sufficient small value, the algorithm terminates, and the best accepted configuration is returned as the solution. As shown in (29), when $S(\pi)$ increases, $f(\pi)$ also increases. Therefore, with high probability, we will draw samples corresponding to $S(\pi)$, which, by design, will maximize the average user PQoS. Next, we design a mechanism for moving from one state to another in the chain. To do so, we define a neighbor of a partition in the sample space Ω as follows.

Definition 5.1: A Partition Policy π_j is Called a Neighbor of a Partition Policy π_i iff π_i and π_j Differ in Only One Element: From the aforementioned definition, π_j can be generated from π_i by replacing an element of π_i with an element drawn randomly from the index set $\mathcal{I} = \{1, 2, \ldots, M\}$. For example, when $M = 5$, partition $\pi_i = (1, 1, 3, 2, 3)$ has a neighbor $\pi_j = (1, 1, 1, 2, 3)$ (because π_i and π_j differ only in the third element).

One should note that, in the context of the MCMC, each partition policy corresponds to a state. We now propose a simulated-annealing-based algorithm to generate samples according to the designed target distribution. We propose a transition function $q(\pi_i, \pi_j)$ that specifies the probability to move from state π_i to one of its neighboring states π_j. Specifically, an element of π_i is selected uniformly at random, and then it is replaced by one of the possible indexes uniformly. Therefore, we have that

$$q(\pi_i, \pi_j) = q(\pi_j, \pi_i) = \frac{1}{2M(M-1)}. \tag{30}$$

Consequently, the acceptance probability, i.e., the probability that the chain moves from the current state π_i to a neighboring state π_j, is given by

$$\begin{aligned}\alpha(\pi_i, \pi_j) &= \min\left\{1, \frac{f(\pi_j)q(\pi_j, \pi_i)}{f(\pi_i)q(\pi_i, \pi_j)}\right\} \\ &= \begin{cases} 1, & \text{if } S(\pi_j) \geq S(\pi_i) \\ e^{\frac{S(\pi_j)-S(\pi_i)}{T_B}}, & \text{if } S(\pi_j) < S(\pi_i). \end{cases}\end{aligned} \tag{31}$$

As defined, the chain will move from the current state π_i to a new state π_j with probability of one, if π_j is a better state (state has higher user satisfaction value). Otherwise, the chain will move to a new state π_j with probability of $e^{(S(\pi_j)-S(\pi_i))/T_B}$. With this design, the whole state space will be explored, if we run the algorithm sufficiently long. Furthermore, the Boltzmann distribution will become increasingly more concentrated around the global maximizer, by gradually decreasing the temperature T_B. Pseudocode of the simulated-annealing-based algorithm is described in Algorithm 1.

Algorithm 1: CARE-SAB Algorithm.

Input: M, c_i, FEC code (n_i, k_i).
Output: Optimal Flow Partition.
1: **STEP 1**: Initialize the starting state π_0 and temperature T_0. Set $n = 0$.
2: **STEP 2**: With probability 1/2, generate a new state π_j from the proposal $q(\pi_n, \pi_j)$.
3: **STEP 3**:
4: **if** $S(\pi_j) \geq S(\pi_n)$ **then**
5: $\quad \pi_{n+1} = \pi_j$
6: **else**
7: $\quad U \sim U(0, 1)$ {Generate a uniform random variable.}
8: $\quad$ **if** $U < \alpha(\pi_n, \pi_j) = e^{\frac{S(\pi_j)-S(\pi_n)}{T_n}}$ **then**
9: $\qquad \pi_{n+1} = \pi_j$
10: $\quad$ **else**
11: $\qquad \pi_{n+1} = \pi_n$
12: $\quad$ **end if**
13: **end if**
14: **STEP 4**: Decrease the temperature $T_{n+1} = \beta.T_n$ where $\beta < 1$, increase n by 1 and repeat from **STEP 2** until stopping condition satisfied (*).
15: **STEP 5**: Return a scheme π that produces the maximum weighted-average satisfaction factor.

Remark (*): The stopping condition can be set by the number of iterations or the difference between the two consecutive states is less than a prespecified value.

B. Upper Bound of Convergence Time

1) Convergence Correctness: The guarantee of convergence to the target distribution using the CARE-SAB algorithm is shown via the following theorem.

Theorem 5.2: Samples drawn from the CARE-SAB algorithm form a Markov chain (MC) whose states satisfy the detailed balance equation AQ3

$$\theta(\pi_i)P(\pi_i, \pi_j) = \theta(\pi_j)P(\pi_j, \pi_i) \quad \forall \pi_i, \pi_j \in \Omega \tag{32}$$

where $\theta(\pi_i)$ and $\theta(\pi_j)$ are the stationary distributions of states π_i and π_j; $P(\pi_i, \pi_j)$ and $P(\pi_j, \pi_i)$ are, respectively, the transition probabilities from state π_i to state π_j and vice versa.

Proof: The proof is provided in the Appendix. ■

The result of Theorem 5.2 shows that samples drawn from the designed algorithm form an ergodic MC, i.e., every state can go to every state; thus, it is possible to find an optimal solution as long as the algorithm runs sufficiently long.

2) Convergence Time: Here, we will derive an upper bound of convergence time to the target distribution. We first define the variation distance at time step k with respect to an initial state π_0 of the MC as

$$\triangle_{\pi_0}(k) \triangleq \max_{\pi \in \Omega} \left| P^k(\pi_0, \pi) - \theta(\pi) \right|. \tag{33}$$

Then, the convergence time of the MC to the target distribution is measured by

$$\tau_{\pi_0}(\varepsilon) \triangleq \min\{k : \triangle_{\pi_0}(k') \leq \varepsilon \quad \forall k' \geq k\} \tag{34}$$

where $0 < \varepsilon$ is an arbitrary infinitesimal value specifying how close the desired solution to an optimal solution. To bound the convergence time at which the chain approaches its stationary distribution (optimal solution), we use the canonical path technique [21]. Letting $e = (\pi_x, \pi_y) \in \Omega^2$, we define $Q(e) \triangleq Q(\pi_x, \pi_y) = \theta(\pi_x)P(\pi_x, \pi_y)$, and a graph $G = (\Omega, E)$, where $(\pi_x, \pi_y) \in E$ *iff* $Q(\pi_x, \pi_y) > 0$. For every ordered pair $(\pi_x, \pi_y) \in \Omega^2$, a canonical path ς_{xy} through G from π_x to π_y is specified by a sequence of legal transitions in G that leads from initial state π_x to final state π_y. Let $\Gamma \triangleq \{\varsigma_{xy} : \pi_x, \pi_y \in \Omega\}$ denote the set of all canonical paths. We now define the edge congestion for the set Γ as follows:

$$\rho(\Gamma) \triangleq \max_{e \in E} \frac{1}{Q(e)} \sum_{\varsigma_{xy} \ni e} \theta(\pi_x)\theta(\pi_y)|\varsigma_{xy}| \tag{35}$$

where $\gamma_{xy} \ni e$ implies that ς_{xy} uses the directed edge e, and $|\varsigma_{xy}|$ denotes the length of the path. The convergence time is bounded by the following proposition.

Proposition 5.1: Let $\mathbf{M}$ be a finite, time-reversible, and ergodic MC over Ω with self-loop probabilities $P(x, x) \geq 1/2$ for all $x \in \Omega$ and stationary distribution π. If the congestion of $\mathbf{M}$ is ρ, then the mixing time of $\mathbf{M}$ satisfies $\tau_x(\epsilon) \leq \rho(\ln \pi(x)^{-1} + \ln \epsilon^{-1})$, for any choice of initial state x.

We are now ready to bound the mixing time of the proposed approximation algorithm CARE-SAB. We choose canonical paths ς_{xy} from any state π_x to any state π_y, which takes T steps, changing π_{x_i} to π_{y_i} on the ith step. Thus, when $\pi_{x_i} = \pi_{y_i}$ for some i, the ith step is just a self-loop. We derive a bound for the mixing time of the chain. Considering any edge $e = (\pi_u, \pi_v)$ for $u \neq v$, we have

$$Q(e) = \theta(\pi_u)P(\pi_u,\pi_v) \overset{(a)}{=} Ce^{\frac{S(\pi_u)}{T_B}} \frac{\min\left\{1, e^{\frac{S(\pi_v)-S(\pi_u)}{T_B}}\right\}}{2M(M-1)}$$
$$= \frac{C}{2M(M-1)} \min\left\{e^{\frac{S(\pi_u)}{T_B}}, e^{\frac{S(\pi_v)}{T_B}}\right\} \overset{(b)}{\geq} \frac{C}{2M(M-1)} \tag{36}$$

where (a) follows from (29)–(31), and (b) follows by using the fact that a partition policy could have a zero-PQoS value. In addition, we have

$$\theta(\pi_u)\theta(\pi_v) = C^2 e^{\frac{S(\pi_u)+S(\pi_v)}{T_B}} \leq C^2 e^{\frac{2}{T_B}} \tag{37}$$

where the last inequality follows from the fact that $S(\pi) \leq 1$ for any partition scheme π. We now compute the number of canonical paths ς_{xy} that use edge e. Note that a canonical path ς_{xy} uses edge $e = (\pi_u, \pi_v)$, where π_u and π_v differ only in the ith element *iff* $\pi_{x_j} = \pi_{u_j}$ for $j = i, \ldots, T$ and $y_j = v_j$ for $j = 0, \ldots, i$. Thus, the number of canonical paths that use edge e is K_m^{T-1}. We now bound the edge congestion as

$$\rho = \max_{e\in\Omega} \frac{1}{Q(e)} \sum_{\varsigma_{xy}\ni e} \pi(x)\pi(y)|\varsigma_{xy}|$$
$$\leq 2CT^2(K_m - 1)(K_m)^{T-1} e^{\frac{2NK_m}{T_B}}. \tag{38}$$

In addition, we have $C = 1/\sum_{x\in\Omega} e^{(S(x))/T_B} \leq 1/|\Omega| = 1/K_m^T$. Therefore

$$\rho \leq 2T^2 e^{\frac{2NK_m}{T_B}}. \tag{39}$$

Using the result from Proposition 5.1 and noting that $\pi(x) \geq 1/|\Omega| = 1/K_m^T$, we then have the convergence time bounded by

$$\tau_x(\epsilon) \leq 2T^2 e^{\frac{2NK_m}{T_B}} \left(\ln K_m^T + \ln \epsilon^{-1}\right). \tag{40}$$

As expected, when the value of ϵ decreases (i.e., closer to the optimal solution), it requires longer runtime. However, as we will show in the simulation, on the order of hundred iterations, the approximation algorithm can obtain a close-optimal solution.

VI. Simulations and Discussions

A. Basic Setup

Network Parameters: We consider a realistic wireless access network having different types of applications with time-varying channel conditions. We first consider a network consisting of five data flows of different applications and compare the performance of different transmission strategies. We then increase the number of data flows to evaluate the robustness of the approximation algorithm. We assume that there are two classes of services: delay-sensitive and elastic traffic. The transmitter decides a coding scheme for a flow, based on the cost at which the user had paid to the service provider, i.e., the higher cost, the higher priority. Note that the service class and priority of a data flow can be easily elaborated in the header of the transmitted packets. In the UNI technique, the transmitter uses the priorities of the incoming flows to assign their redundancies, and they will be used in all the techniques for a fair comparison. We consider a wireless channel with a bandwidth of 2 Mb/s, which is equivalent to $N = 133$ time slots or 133 1.5-kB packets. In addition, an elastic traffic requires 18 data packets per second, corresponding to a rate of 27 kb/s, while a delay-sensitive traffic requires 25 and 30 data packets, corresponding to rates of 37.5 and 45 kb/s, for medium and high QoS, to achieve a PQoS value of one. Our parameters are set based on the number of frames per second in video streaming and the standard service specifications [48]. For example, our delay-sensitive flow can be used to model a Voice over IP (VOIP) call (e.g., using G.728 standard with a codec interval of 5 (ms) requires a transmission rate of 31.5 kb/s [48]).

TABLE I
Parameters of the Incoming Flows

Flow ID	Rx ID	FEC (n_i, k_i)	p_i	Service type	Priority
$f1$	D_1	$(21, 18)$	–	Elastic	1
$f2$	D_2	$(21, 18)$	0.05	Elastic	1
$f3$	D_3	$(22, 18)$	0.05	Elastic	2
$f4$	D_4	$(31, 25)$	0.05	Delay-sensitive	3
$f5$	D_5	$(38, 30)$	0.05	Delay-sensitive	4

If the number of data packets received at each receiver is less than the required packets, its satisfaction will decrease in accordance to the PQoS functions, as described in Section III-C. The transmission parameters of the incoming flows are given in Table I. These parameters are set based on the types of applications, priorities of the incoming flows, and the bandwidth availability. In addition, the redundancy used for each incoming flow depends on its priority; for example, in our experiments, we set priorities 1, 2, 3, and 4, corresponding to redundancies of 15%, 20%, 25%, and 30%, respectively. Note that these parameters will be applied to all techniques.

Transmission Strategies: We evaluate the performance of the following transmission strategies for comparison.

- *Unicast (UNI)*: The UNI scheme transmits data of the incoming flows separately, without using NC.
- *Random Network Coding (RNC)*: RNC scheme implements a standard interflow NC technique, where data packets of all flows are combined together using RNC [41]. The coded packets are then broadcasted to all the receivers.
- *Systematic Random Network Coding (SRNC)*: SRNC implements a simple systematic NC, where original data packets are transmitted in the first phase and coded data packets (generated by using RNC across all data flows) are transmitted in the second phase.
- *Type of Flow (ToF)*: The first naïve ToF scheme mixes all data of flows with the same application types. Such a transmission strategy does not have to perform intensive optimization computation; however, it might limit the effect of mismatched mixing of the NC-based schemes, by combining data of flows with the same types.

- *Priority of Flow (PoF) Scheme*: Differently, the PoF scheme selects flows that have the same priority as subgroups to perform NC. The idea of using the PoF scheme is to utilize the priority factors of the incoming data flows to quickly classify them into different subgroups for NC. Intuitively, encoding data of the same priority flows could enhance the QoS of the network.
- *Network Coding for Throughput (NCT)*: The NCT scheme is simulated based on the work in [5], which is proposed for maximizing the network throughput. In NCT, the sender considers the packet at the head of its queue as a primary packet and selects side packets to construct coded packets so that it maximizes the number of possible receivers at the current time slot.
- *CARE/CARE-EXH*: This scheme uses the optimal mixing solution via exhaustive search for data transmission.
- *CARE-SAB*: CARE-SAB uses an approximation algorithm based on the proposed CARE-SAB algorithm for data encoding and scheduling.
- *2-PCARE*: Based on the PCARE scheme described in Section IV-D, we simulate 2-PCARE for comparison. In this scheme, each incoming flow is divided into two equal subflows, and then, these subflows are optimally combined for transmission.

B. Data Recovery

We first show the benefit of *informed* mixing and the drawback of blind mixing by examining the probability that all the receivers can decode their packets using strategies derived in Section IV. Fig. 5(a) shows the probability of data recovery versus partition policies, i.e., the way of mixing data when packet losses of receivers from D_2 to D_5 are set to 5% while that of receiver D_1 is 13%. We map each partition policy to an integer on the x-axis. The number of possible partition policies is an exponential function of M, and this is equal to the sum of the Stirling numbers of the second kind, as shown in (28). We also plot the recoverability probabilities for UNI and SRNC techniques on the same graph for comparison. They are indicated by straight lines since these techniques do not depend on the partition policies. Recall that UNI does not mix packets from different flows. SRNC sends the original packets and then the mixed redundant packets; hence, the amount of mixing here is rather minimal. As observed, SRNC is clearly better than UNI. It is interesting to note that, at least in this scenario, blind mixing is generally better than UNI. As expected, CARE-SAB results in different recoverability probabilities, depending on which flows are combined with each other. In our program, we let the CARE-SAB algorithm run until converged. Based on our deeper data analysis generated by the program, the proposed CARE-SAB finds the best partition by mixing flows f_1 and f_4 into one group and flows f_2, f_3, and f_5 into another group.

Next, we evaluate the data recovery probability by all receivers versus the packet loss probability in Fig. 5(b). In this scenario, the packet loss probabilities of receivers $D_i, i = \{2, \ldots, 5\}$ are shown in Table I, while the packet loss rate of receiver D_1 is varied from 1% to 22%. As expected, CARE-SAB outperforms the other schemes, with considerable gaps,

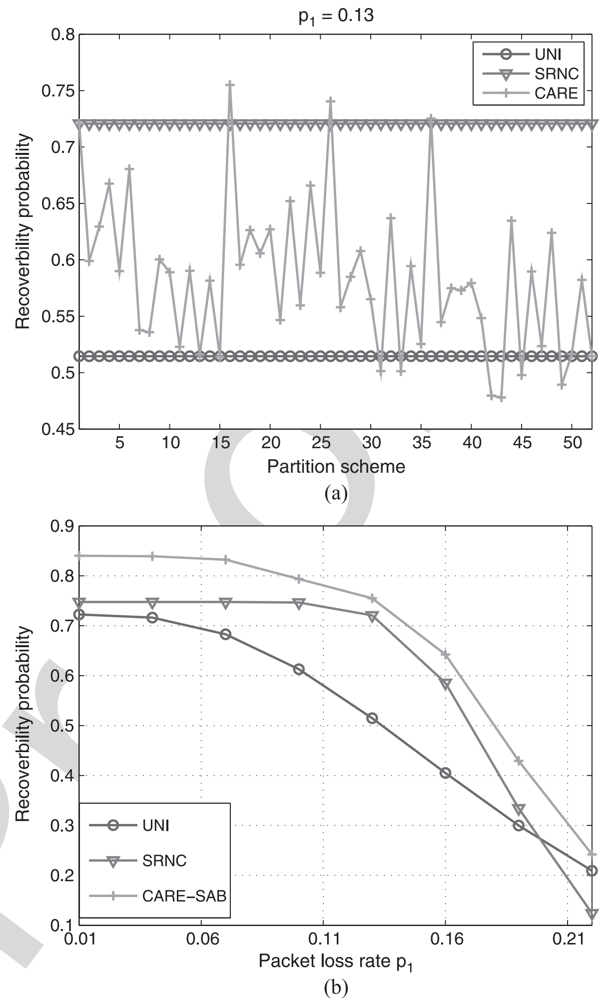


Fig. 5. Recoverability probability versus (a) partition scheme and (b) packet loss rate p_1.

due to its selective mixing of the flows. In addition, we observe that, when p_1 is less than 18%, SRNC achieves better performance than UNI. In other words, in this erasure regime, mixing data packets across all flows would be more beneficial than transmitting them separately. However, this is not the case when the packet loss p_1 is greater than 18%. In such a setting, the UNI scheme outperforms SRNC. The intuition is that SRNC combines the data of all the flows; as a result, each receiver needs to obtain at least a full set of coded packets to recover its data. However, this may not be possible to receiver D_1 because it experiences a deep fading, leading to substantial reduction on the overall network recoverability. In such a case, separately transmitting data to different users will be a better option.

C. User's PQoS Versus Erasure Probability

We first evaluate the individual PQoS versus the transmission channel conditions. In particular, we set $p_3 = p_4 = 5\%$, $p_2 = p_1 + 0.01$, and $p_5 = p_1 + 0.02$. The other parameters of the network are set the same as before in Table I. The base values of the PQoS, i.e., γ_0, of the delay-sensitive and elastic traffic are set at 0.5 and 0.6, when the number of useful packets received equals 50% of the intended packets. This setting is to reflect that delay-sensitive applications are more vulnerable to AQ4

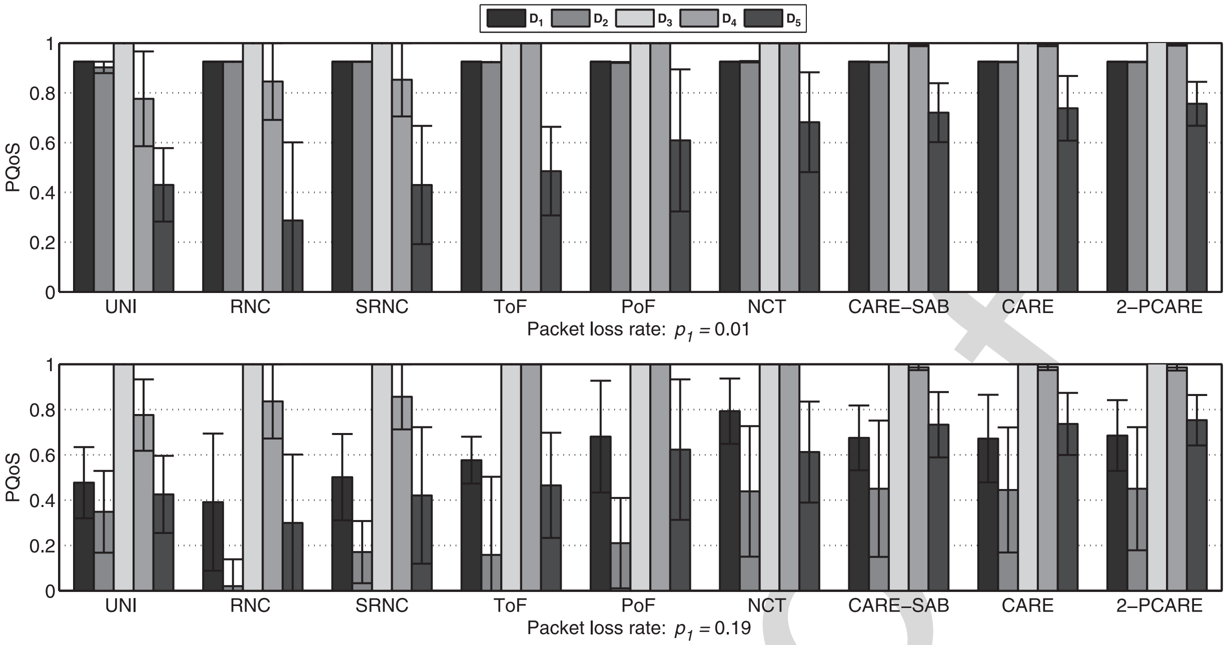


Fig. 6. PQoS values of users in different packet error regimes (order of the receivers is the same, as illustrated in the legend).

packet losses than the elastic traffic. The CARE-SAB algorithm runs for 30 iterations, using the normalization factor $C = 1$, initial temperature $T_0 = 1$, and cooling scale factor $\beta = 0.9$. We use the simplest partition policy $k = 2$ for the PCARE scheme, where each flow is evenly divided into two subflows. We perform many trials and compute the average of the results.

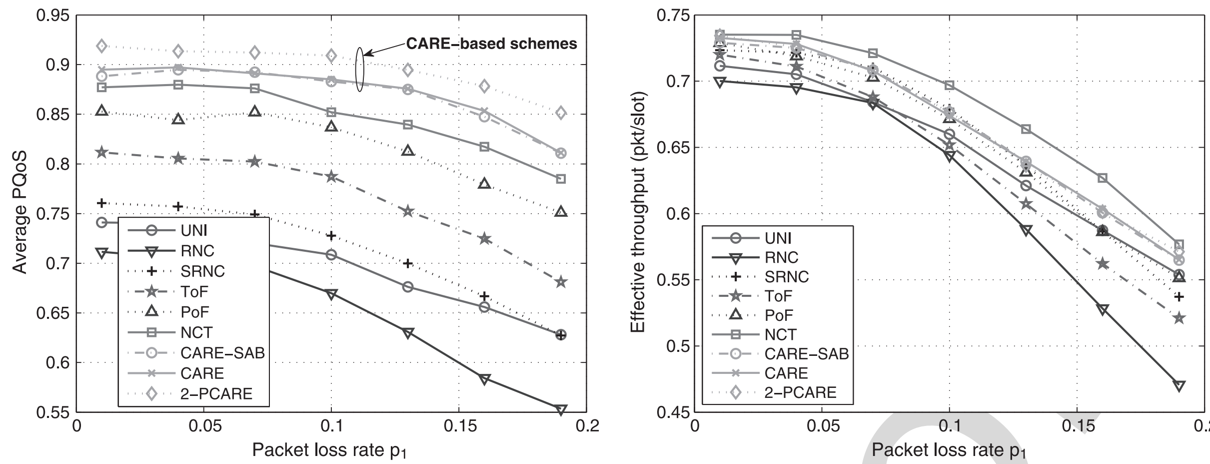
Fig. 6 represents the PQoS values across all users in two different regimes of the packet erasure probabilities. In the low-packet-loss regime (upper part in Fig. 6), i.e., $p_1 = 0.01$, all the strategies satisfy the QoS of the first four flows. This is intuitively plausible since, in this case, transmission bandwidth is plenty for these flows, no optimization is needed, and all these users get what they want. However, the RNC scheme significantly decreases the PQoS of D_5. This is because, when combining all the data packets together, it cannot recover the transmitted data, resulting in a degraded PQoS. On the other hand, the CARE-based schemes that balance data types and priorities for different groups achieve the best performance. Furthermore, the 2-PCARE scheme with a finer grained data partition policy achieves the best performance.

On the other hand, in the high-packet-loss regime (lower part in Fig. 6), i.e., $p_1 = 0.19$, all the receivers with low packet loss rates, i.e., D_3 and D_4, can still maintain a high PQoS in all transmission strategies. However, the PQoS of the receivers with higher packet loss rates, i.e., D_1, D_2, and D_5, significantly decreases in all strategies. In particular, receiver D_2 has its PQoS decreased substantially in the SRNC scheme. The intuition is that mixing up data of all flows makes it difficult for D_2 to receive a full set of the coded packets in the high-erasure-probability regime. As a result, D_2 is not able to recover its data. As expected, CARE (i.e., exhaustive search) that searches all the possibilities of partition policies always achieves the best performance. However, an interesting observation is that the CARE-SAB algorithm can approximate the optimal solution with only 30 iterations. This significantly reduces the search time compared with the exhaustive search in larger size problems. Indeed, the PQoS achieved by CARE-SAB is very close to that of the exhaustive search CARE, with a marginal gap.

D. Network PQoS and Effective Throughput Versus Erasure Probability

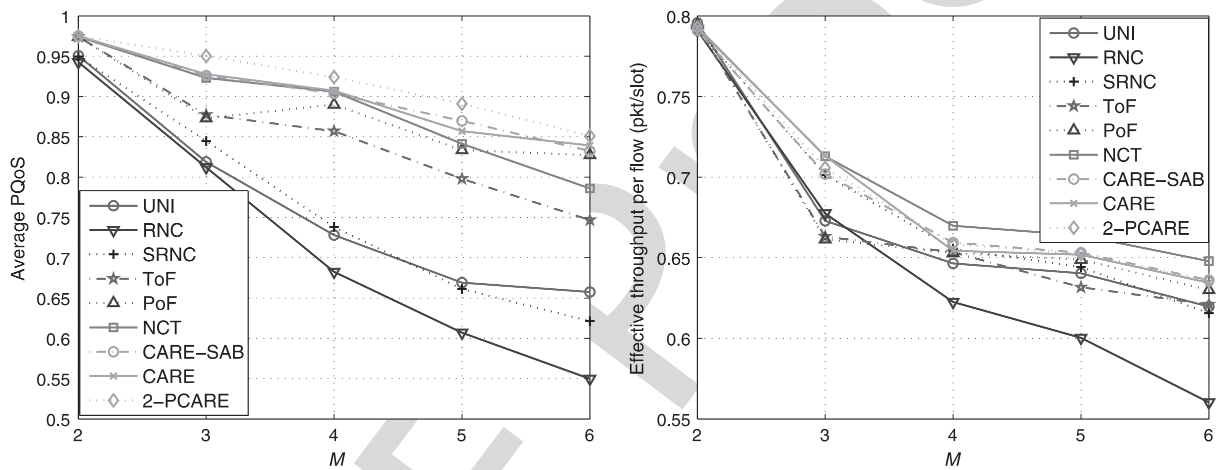
Next, we compare the network PQoS and effective throughput of different transmission strategies versus the packet loss p_1. The effective throughput is computed based only on the received data, contributing to the QoS of the users, without considering application types. Fig. 7(a) shows the average PQoS across all receivers. As expected, the average PQoS decreases with the increase of p_1. We observe that RNC has the worst PQoS out of all strategies. This is because of the degraded PQoS of receivers that cannot recover the transmitted data due to bad channel conditions. Interestingly, UNI outperforms RNC with a significant performance gap, due to its partially recovered data. On the other hand, in RNC, the receivers need to receive a full set of coded packets for data decoding; however, this may not be possible due to poor transmission channel conditions. The NCT outperforms the other schemes but less than that of the CARE-based schemes, as shown in Fig. 7(a). This is because greedily optimizing the throughput without considering the characteristics of the traffic could significantly decrease the network QoS. Again, 2-PCARE achieves the best performance, which is followed by CARE-based schemes. We further observe that CARE-SAB achieves an identical performance of CARE (i.e., exhaustive search) with much less runtime. This confirms the efficiency and robustness of the proposed CARE-SAB algorithm.

Next, we investigate the network effective throughput versus the packet error rate in Fig. 7(b). As expected, NCT achieves the best performance because its objective is to maximize the



(a) (b)

Fig. 7. Network performance of different schemes versus p_1 with $M = 5$: (a) average PQoS and (b) effective network throughput.



(a) (b)

Fig. 8. Network performance of different schemes versus M for (a) average PQoS and (b) effective network throughput.

network throughput. Interestingly, CARE-based schemes not only outperform the other techniques but also approximate to NCT, despite that the objective of CARE is to maximize the average PQoS. The explanation is as follows. First, CARE uses PQoS in its formulation, whereby the value of PQoS is computed based on both the effective throughput (i.e., number of useful packets) and characteristics of the information flows. Second, and more importantly, PQoS functions are well designed, so that a small change in effective throughput is transformed into the users' PQoS. Thus, optimizing the PQoS results in a near-optimum network effective throughput. We additionally observe that the RNC scheme suffers from combining the data of all flows, resulting in the worst performance, particularly in the presence of deep channel fading.

E. PQoS and Throughput Versus Number of Flows

We next examine the overall network PQoS and effective throughput versus the number of data flows in Fig. 8(a) and (b). As expected, the CARE-based schemes outperform the others with considerable gaps. The RNC scheme achieves the worst performance and significantly decreases as M increases. This is because all-flow encoding suffers from the curse of "all or nothing" of the RNC decoding constraint, i.e., it requires a full set of encoded packets for data recovery. SRNC outperforms UNI in the regime of lower packet error rate, while significantly decreasing with the increase of p_1. This is because encoded packets cannot be recovered due to high lost packets. Interestingly, the two heuristic ToF and PoF schemes achieve better performance compared with the traditional NC-based and UNI schemes. However, when M increases, their performance starts decreasing considerably. As expected, the 2-PCARE scheme achieves the best performance, due to its finer grained subflow partition and encoding. We also observe that the CARE-SAB obtains a competitive performance by using only 30 iterations.

Additionally, Fig. 8(b) illustrates the network effective throughput versus M. As expected, the NCT that is optimized for the throughput achieves the best performance. CARE-based schemes outperform the other techniques, despite that the objective of CARE is to maximize PQoS. This is because PQoS is constructed from both the effective throughput (i.e., number of useful packets) and characteristics of the information flows. Thus, optimizing the PQoS will result in high network effective throughput. The RNC achieves the worst network throughput,

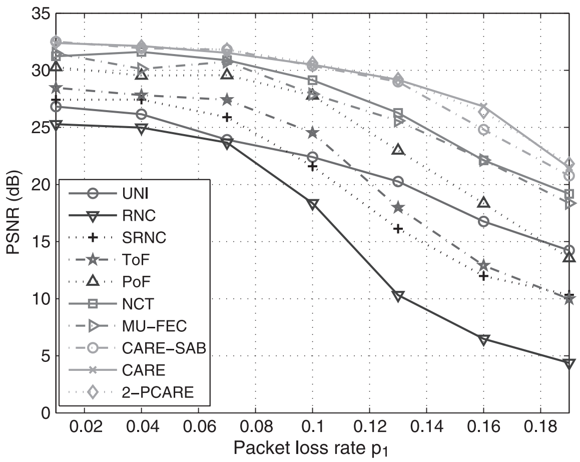


Fig. 9. Average PSNR of video sequences at receivers versus packet lost rate p_1.

and its performance decreases significantly with the increase of M. This is consistent with the intuition that when M increases it is more difficult for the receivers to receive a full set of encoded data for decoding.

F. PSNR Versus Erasure Probability

In addition, we also evaluate the average PSNR based on the luminance (Y) component of *Foreman* and *Coastguard* video sequences. We assume that each frame of the video sequences is packetized into an independent network abstraction layer of 1500 bytes. Furthermore, in our simulation, we use short video sequences, with the *Foreman* and *Coastguard* having 30 and 25 packets, respectively. The longer video sequences can be constructed by concatenating multiple frames. Additionally, we assume that the PSNR of the encoded sequences *Foreman* and *Coastguard* are 29.95 dB [15] and 45.72 dB [6], respectively, corresponding to no-error transmission of those streams.

Fig. 9 illustrates the average PSNR of the two video sequences using different transmission strategies. In our simulation, the transmitter transmits five data flows, consisting of three elastic and two delay-sensitive flows, to five receivers. To compute PSNR, we extract only the received packets of the two video sequences. As expected, when the packet lost rate is small, all schemes perform well with high PSNR. This is because only some of the transmitted data packets were lost. However, when the packet lost rate increases, the video quality at the receivers rapidly degrades. As expected, the proposed CARE-based strategies achieve the highest video qualities, due to their content-aware encoding and scheduling. We also simulated the MU-FEC scheme proposed in [49] for comparison. The MU-FEC scheme exploits intra- and interflow NC for mixing data of different flows at the transmitter to improve bandwidth efficiency. In spite of optimizing its coding for maximizing the network throughput, it does not consider the AQ5 content of the traffic, resulting in low PSRN. This is the same observation for the NCT scheme, which focuses on throughput maximization instead of network QoS. Interestingly, we observe that the heuristic PoF scheme achieves very good performance in terms of PSNR by combining data of only higher priority flows associated with the video sequences. In the regime of high erasure rate, with a smaller encoding data batch, it can successfully transmit data to the receivers with higher probability. On the other hand, RNC combining all data together for transmission suffers because most of the data cannot be recovered at the receivers.

G. CARE-SAB Convergence Rate

We now evaluate the effectiveness and robustness of the CARE-SAB algorithm. In these experiments, we consider data traffic consisting of eight data flows, where the first five flows are the same as before (in Table I), and the additional flows include one elastic and two delay-sensitive flows that are a copy of D_2 and D_4, respectively. We have a total of more than 4×10^3 possible flow partition policies. To illustrate the convergence of the proposed CARE-SAB, we vary the channel conditions randomly with the packet loss rates in the range between 1% and 20%. In the CARE-SAB algorithm, the initial state is set by grouping all flows together. Fig. 10(a) and (b) illustrates a snapshot of the PQoS values changing with respect to the iteration and actual runtime, respectively. The CARE-EXH achieves the best performance by using exhaustive search for all possible partition policies and selects the best partition. On the other hand, CARE-SAB schemes implement the proposed approximation method to find the optimal partition policy. The CARE-SAB-Iter represents state by state the chain visits in AQ6 each iteration, whereas the CARE-SAB records the maximum PQoS values, which have been obtained up to that iteration. The CARE-SAB schemes first aggressively "explore" states, even the ones with low value of PQoS, and gradually "cool" down to the optimal solution. As illustrated in Fig. 10(a), it takes about 200 iterations (about 5% of the search space) for the CARE-SAB schemes to "hit" the optimal solution, indicated by the PQoS merged to that of the CARE-EXH. The simulation result is consistent with the theoretical analysis of the Theorem 5.2, i.e., setting the number of iterations sufficiently large, an arbitrarily near-optimal solution can be obtained via the proposed CARE-SAB algorithm.

We further measure the actual runtime of the algorithm based on the elapsed time of the algorithm implemented on our laptop (OS Window 7, Intel Core i5 with 4-GB RAM). As illustrated in Fig. 10(b), it takes about 0.46 s for the CARE-SAB schemes to find the optimal solution for the case of $M = 8$ flows. We note that this is only one snapshot of a trial. In practice, the runtime could be much less than that, if we eliminate the effect of other concurrent processes in the machine.

We further evaluate the convergence robustness of the proposed CARE-SAB by using different values of β. We recall that β is the parameter controlling the "cooling" process of the search algorithm. Fig. 11(a) illustrates the convergence of CARE-SAB-Iter for different values of β in [0.9, 0.99]. As observed, it requires more time for the system with smaller values of β to converge to the optimal solution. This is because the system with smaller values of β (i.e., $\beta \times T_n$ decreases faster) "jumps" with larger steps at the beginning of the process that may visit several "bad" states before converging to the optimal solution. On the other hand, with greater values of

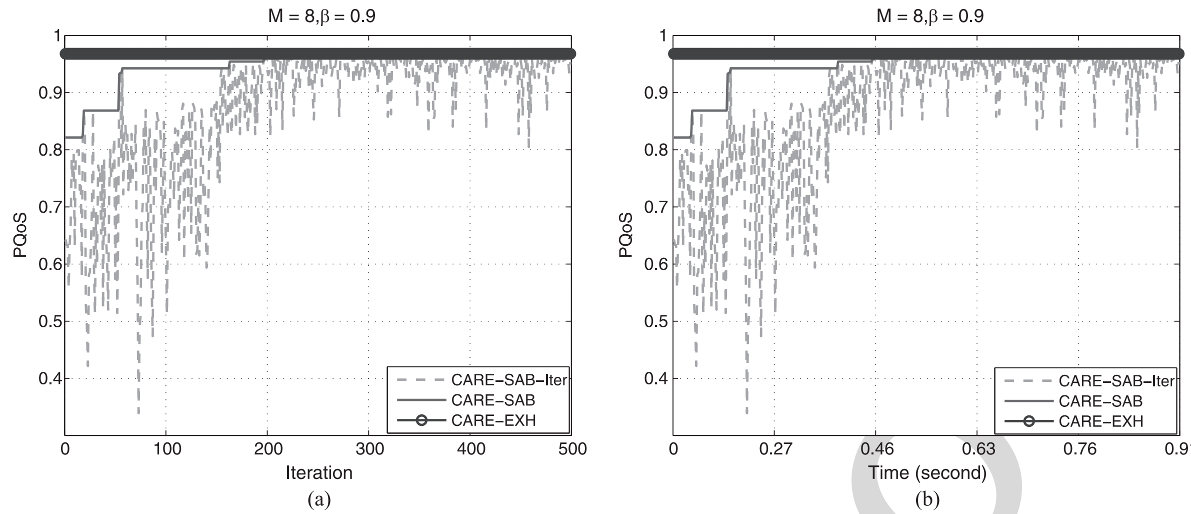


Fig. 10. CARE-SAB convergence versus (a) iteration and (b) time (in seconds): $M = 8$, $T_0 = 100$, and $\beta = 0.9$.

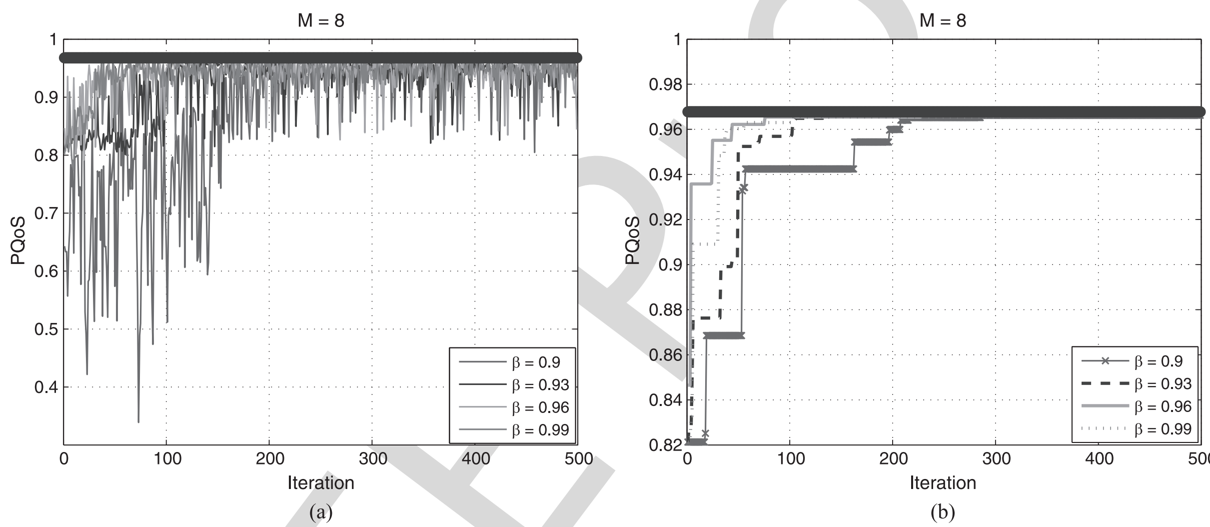


Fig. 11. CARE-SAB convergence for different values of β, $M = 8$, $T_0 = 100$.

β, the system is more conservative in giving high probability for exploring "good" states. This setting could reduce the "burning" time of the process before convergence, but it may result in a local optimal solution. Our experiments illustrated in Fig. 11(a) shows that the system quickly converges to the optimal solution within 300 iterations, for all implemented values of β. Fig. 11(b) illustrates the maximal value of PQoS that has been found up to the current iteration. Additionally, the result shows that, for some case, e.g., $\beta = 0.99$, the algorithm quickly obtains the optimal solution in only 100 iterations (about 2.5% of the search space).

Similar performance is obtained for the case of $M = 10$, as illustrated in Fig. 12. As we can observe, the proposed algorithm CARE-SAB quickly converges to the optimal solution and is robust with respect to the values of β. For some cases, e.g., $\beta = 0.9$, the CARE-SAB needs more iterations to find the optimal solution (about 450 iterations). However, we should note that, when $M = 10$, there could have 115 975 possible partition policies. Therefore, the increase is justifiable compared with the exponential increase of the search space.

Additionally, we further evaluate the convergence rate of the proposed CARE-SAB with different values of the initial temperature T_0. Fig. 13(a) and (b) illustrates the convergence of CARE-SAB-Iter and CARE-SAB for $M = 8$ and $\beta = 0.9$, respectively. As illustrated, the proposed algorithm is robust with respect to different initial values of T_0. We further observe that initializing T_0 with different values only slightly affects the system convergence rate. This is an expected result and consistent with the theoretical analysis because the temperature controls how the algorithm explores the search space. In particular, greater value of T_0 provides more freedom to the algorithm to explore more states, resulting in longer convergence time. However, such a setting will ensure that the global optimal solution will be "hit" with high probability. Similar results are also obtained for the case of $M = 10$ flows, as illustrated in Fig. 14. With larger search space, in the worst case ($T_0 = 100$), the algorithm requires about 400 iterations to find the optimal solution.

Finally, we evaluate the scalability and efficiency of the proposed CARE-SAB in Fig. 15. In particular, Fig. 15(a) AQ7 compares the performance of the CARE-SAB using different

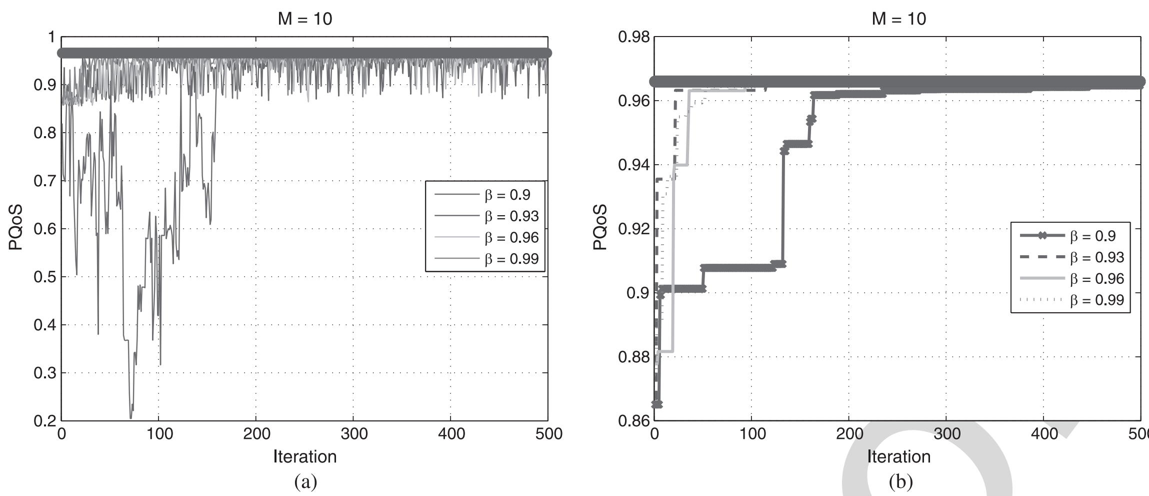


Fig. 12. CARE-SAB convergence for different values of β, $M = 10$, $T_0 = 100$.

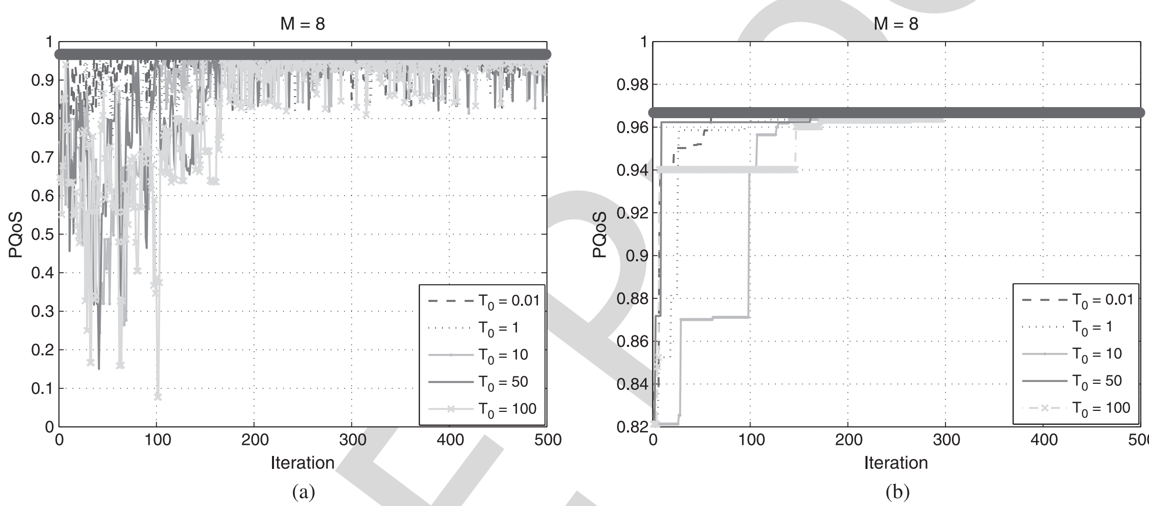


Fig. 13. CARE-SAB convergence for different values of T_0, $M = 8$, $\beta = 0.9$.

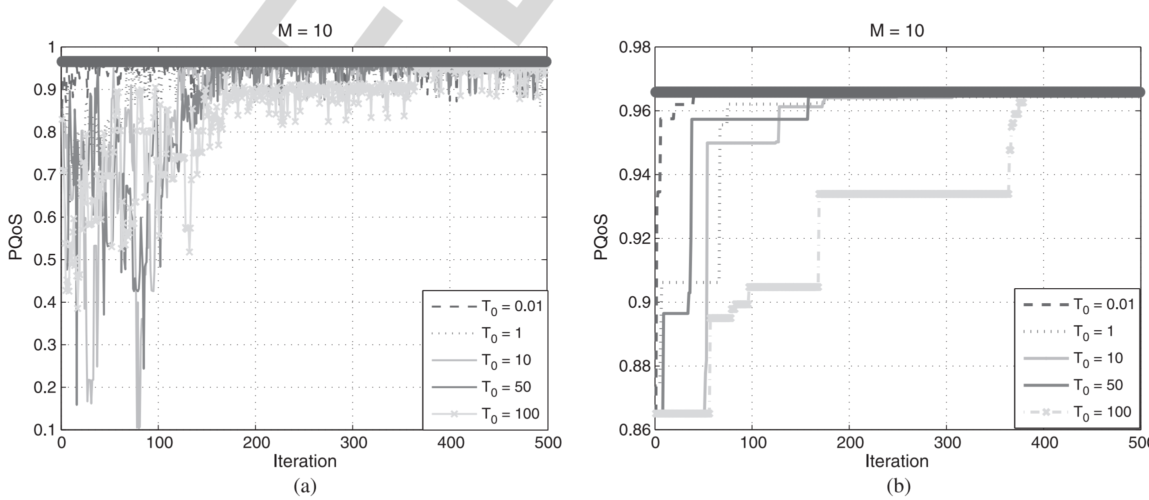
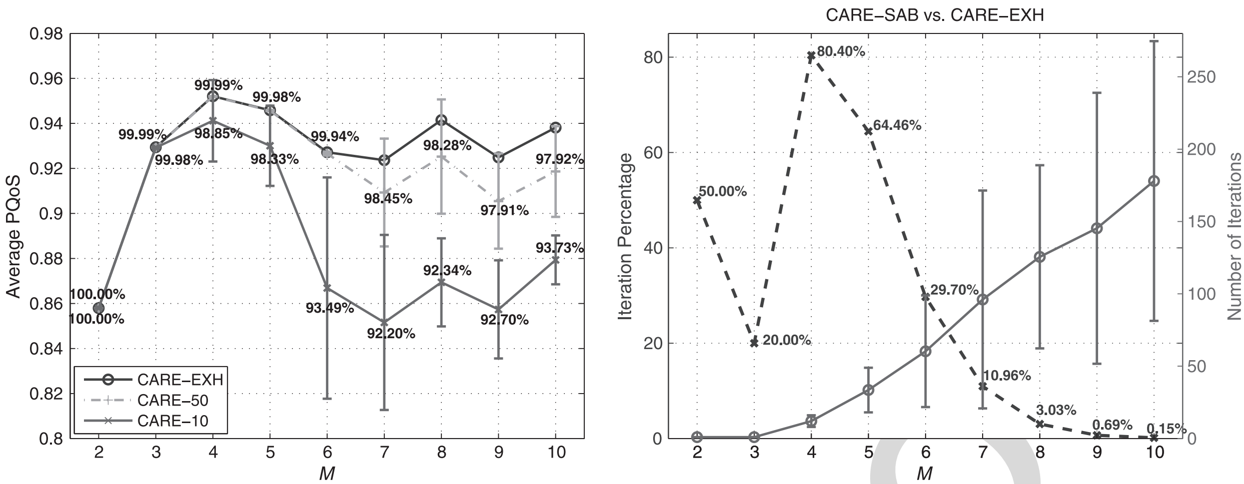


Fig. 14. CARE-SAB convergence for different values of T_0, $M = 10$, $\beta = 0.9$.

values of iterations with the exhaustive search CARE-EXH versus M, where CARE-10 and CARE-50 represent CARE-SAB that uses 10 and 50 iterations, respectively. As expected, when M increases and given a fixed number of iterations, the performance of CARE-SAB schemes decreases due to the larger search space. However, it is interesting to observe that CARE-SAB schemes can achieve about 90% performance of the exhaustive search despite using a fixed number of iterations. The results illustrate that the designed algorithm is scaled well with the problem size.



(a) (b)

Fig. 15. Evaluating the scalability and robustness of CARE-SAB with $T_0 = 1$ and $\beta = 0.9$: (a) average PQoS for different numbers of iterations and (b) number of iterations for CARE-SAB to obtain optimal partition policy and percentage compared with CARE-EXH.

We further evaluate the scalability of CARE-SAB algorithm by changing M in Fig. 15(b). In particular, we compute the average number of iterations that the CARE-SAB algorithm consumes to find an optimal solution. In our experiment, for each M, we run the algorithm many times and compute the average of the results. Fig. 15 illustrates the average number of iterations of CARE-SAB and the corresponding percentage compared with CARE-EXH for $M = 2, \ldots, 10$. As expected, the number of iterations of CARE-SAB (i.e., the red line) increases with M, due to the exponential increase of the search space. However, compared with the exhaustive search CARE-EXH, the number of states that CARE-SAB explores to find the optimal solution decreases significantly (i.e., the dash blue curve), i.e., from 50% for $M = 2$ to 3.03% for $M = 8$ and to 0.15% for $M = 10$. The simulation results also confirm that the proposed CARE-SAB algorithm is also scaled well with M.

VII. Conclusion

We have investigated the problem of mixing data of traffic with different service classes to improve the network QoS. We first showed that exhaustively mixing data across different data flows at every opportunity may substantially decrease the network QoS. We then proposed CARE, a context-aware interflow network coding and scheduling, to maximize the QoS across all the receivers based on the user satisfaction PQoS. The objective function of CARE is formulated by considering not only the characteristics of traffic but also the service classes and channel conditions. We then showed the hardness of finding an optimal solution to CARE and proposed an efficient approximation algorithm, i.e., CARE-SAB, to obtain a guaranteed near-optimal solution. We further proved the correctness and derived an upper bound on the convergence time of the CARE-SAB algorithm. In addition, we described the PCARE scheme that partially combines data of different flows to further improve the network performance. Simulation results showed that up to a 50% performance gain of the proposed CARE-based schemes can be achieved compared with the existing approaches (e.g., RNC). The results also showed that the approximation algorithm is robust with respect to the heuristic parameters and well scaled with the number of data flows. To the best of our knowledge, this work is one of a few works studying NC from the QoS point of view. One of our future extensions is to investigate an efficient algorithm for optimal subflow partitioning and encoding of the PCARE scheme.

Appendix
Proof of Theorem 5.2

Let us consider any two states $\pi_i, \pi_j \in \Omega$. According to the proposed target distribution, we have $\theta(\pi_i) = Ce^{(S(\pi_i))/T_B}$ and $\theta(\pi_j) = Ce^{(S(\pi_j))/T_B}$. Therefore, we have two possibilities.

Case 1: If $S(\pi_i) \le S(\pi_j)$, we first consider the direction moving from state π_i to state π_j. From (31), we have $\alpha(\pi_i, \pi_j) = 1$. Thus

$$\theta(\pi_i)P(\pi_i, \pi_j) = Ce^{\frac{S(\pi_i)}{T_B}} q(\pi_i, \pi_j). \qquad \text{(A.1)}$$

For the direction from state π_j to state π_j, we have

$$\alpha(\pi_j, \pi_i) = e^{\frac{S(\pi_i) - S(\pi_j)}{T_B}}. \qquad \text{(A.2)}$$

Hence

$$\begin{aligned} \theta(\pi_j)P(\pi_j, \pi_i) &= Ce^{\frac{S(\pi_j)}{T_B}} q(\pi_j, \pi_i)\alpha(\pi_j, \pi_i) \\ &= Ce^{\frac{S(\pi_j)}{T_B}} q(\pi_j, \pi_i) e^{\frac{S(\pi_i) - S(\pi_j)}{T_B}} \\ &= Ce^{\frac{S(\pi_i)}{T_B}} q(\pi_j, \pi_i). \qquad \text{(A.3)} \end{aligned}$$

Since $q(\pi_i, \pi_j) = q(\pi_j, \pi_i)$, therefore, from (A.1) and (A.3), we obtain the detailed balance equation.

Case 2: Now consider the scenario where $S(\pi_i) > S(\pi_j)$. Similarly, from (31), we have

$$\alpha(\pi_i, \pi_j) = e^{\frac{S(\pi_j) - S(\pi_i)}{T_B}}$$
$$\alpha(\pi_j, \pi_i) = 1.$$

Thus

$$\begin{aligned}\theta(\pi_i)P(\pi_i,\pi_j) &= Ce^{\frac{S(\pi_i)}{T_B}}q(\pi_i,\pi_j)\\ &= Ce^{\frac{S(\pi_i)}{T_B}}q(\pi_j,\pi_i)e^{\frac{S(\pi_j)-S(\pi_i)}{T_B}}\\ &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i) \qquad (A.4)\\ \theta(\pi_j)P(\pi_j,\pi_i) &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i)\alpha(\pi_j,\pi_i)\\ &= Ce^{\frac{S(\pi_j)}{T_B}}q(\pi_j,\pi_i). \qquad (A.5)\end{aligned}$$

From (A.4) and (A.5), we have the detailed balance equation; therefore, the theorem follows. ∎

REFERENCES

[1] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 6, pp. 1204–1216, Jul. 2000.

[2] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast with network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.

[3] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On delay performance gains from network coding," in *Proc. 40th Annu. Conf. Inf. Sci. Syst.*, 2006, pp. 864–870. AQ8

[4] T. Ho, M. Medard, M. Effros, and D. R. Karger, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Symp. Inf. Theory*, 2003, p. 442.

[5] S. Katti *et al.*, "XORs in the air: Practical wireless network coding," in *Proc. ACM SIGCOMM*, 2006, pp. 243–254. AQ9

[6] D. Nguyen, T. Nguyen, and X. Yang, "Wireless multimedia transmission with network coding," in *Proc. Packet Video Workshop*, 2007, pp. 326–335.

[7] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Res., Tech. Rep. MSR-TR-2004-78, 2004.

[8] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," in *Proc. IEEE Inf. Theory Workshop*, 2004, pp. 304–309.

[9] J. Widmer, C. Fragouli, and J.-Y. Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *Proc. Workshop Netw. Coding, Theory, Appl.*, 2005, pp. 1–6.

[10] S. R. Chandran and S. Lin, "Selective-repeat-ARQ schemes for broadcast links," *IEEE Trans. Commun.*, vol. 40, no. 1, pp. 12–19, Jan. 1992.

[11] J. L. Wang and J. A. Silvester, "Performance optimization of the go-back-N ARQ protocols over broadcast channels," *Comput. Commun.*, vol. 14, no. 7, pp. 393–404, Sep. 1991.

[12] T. Tran, T. Nguyen, and B. Bose, "A joint network-channel coding technique for single-hop wireless networks," in *Proc. 4th Workshop Netw. Coding, Theory, Appl.*, 2008, pp. 1–6.

[13] T. Tran, T. Nguyen, B. Bose, and V. Gopal, "A hybrid network coding technique for single-hop wireless networks," *IEEE J. Sel. Topics Adv. Commun.*, vol. 27, no. 5, pp. 685–698, Jun. 2009.

[14] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," in *Proc. Packet Video Workshop*, 2007, pp. 191–200.

[15] H. Seferoglu and A. Markopoulou, "Video-aware opportunistic network coding over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 713–728, Jun. 2009.

[16] J. Domzal, J. Dudek, P. Jurkiewicz, L. Romanski, and R. Wojcik, "The cross-protect router: Implementation tests and opportunities," *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 115–123, Sep. 2014.

[17] J. W. Roberts, "Internet traffic, QoS, and pricing," *Proc. IEEE*, vol. 92, no. 9, pp. 1389–1399, Sep. 2004.

[18] G. Miao and Z. Niu, "Bandwidth management for mixed unicast and multicast multimedia flows with perception based QoS differentiation," in *Proc. IEEE ICC*, 2006, pp. 687–692.

[19] T. Tran and T. Nguyen, "Adaptive network coding for wireless access networks," in *Proc. 19th ICCCN*, Aug. 2010, pp. 1–6.

[20] G. Perboli, R. Tadei, and L. Gobbato, "The multi-handler knapsack problem under uncertainty," Interuniversity Res. Cent., Montreal, QC, Canada, Rep. CIRRELT-2012-69, 2012.

[21] A. Sinclair, "Improved bounds for mixing rates of Markov chains and multicommodity flow," in *Combinatorics, Probability and Computing*. Berlin, Germany: Springer-Verlag, 1992.

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

[23] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[24] M. Medard, M. Effros, T. Ho, and D. Karger, "On coding for non-multicast networks," in *Proc. Allerton Conf. Commun.*, 2003, pp. 1–7.

[25] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, 2003, pp. 1–10. AQ10

[26] T. Ho *et al.*, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[27] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. 43rd Annu. Allerton Conf. Commun.*, 2005, pp. 1–10.

[28] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: An instant primer," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[29] S. Deb *et al.*, "Network coding for wireless applications: A brief tutorial," in *Proc. IWWAN*, 2005, pp. 196–200.

[30] A. Douik, S. Sorour, M. Alouini, and T. Al-Naffouri, "Delay minimization for instant decodable network coding in persistent channels with feedback intermittence," *ArXiv e-Prints*, 2013.

[31] A. Douik, S. Sorour, M. Alouini, and T. Al-Naffouri, "Delay reduction in lossy intermittent feedback for generalized instantly decodeable network coding," in *Proc. IEEE Int. Conf. WiMob Comput., Netw. Commun.*, 2013, pp. 388–393.

[32] R. Koetter and M. Medard, "Beyond routing: An algebraic approach to network coding," in *Proc. IEEE INFOCOM*, 2002, pp. 122–130.

[33] Default TTL Values in TCP/IP, 2013. [Online]. Available: http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm

[34] IP Option Numbers [RFC791], [RFC1122], 2013. [Online]. Available: http://www.iana.org/assignments/ip-parameters/ip-parameters.xml

[35] I. Hou and P. R. Kumar, "Scheduling heterogeneous real-time traffic over fading wireless channels," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[36] M. Mazzotti *et al.*, "Analysis of packet-level forward error correction for video transmission," in *Proc. IEEE VTC Spring*, 2011, pp. 1–5.

[37] C. Feng and B. Li, "Network coding for content distribution and multimedia streaming in P2P networks," in *Network Coding: Fundamentals and Applications*. San Diego, CA, USA: Academic, 2012.

[38] S. Sorour, A. Douik, S. Valaee, T. Y. Al-Naffouri, and M.-S. Alouini, "Partially blind instantly decodable network codes for lossy feedback environment," *IEEE Trans. Wireless Commun.*, vol. 13, no. 9, pp. 4871–4883, Sep. 2014.

[39] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Proc. IEEE Inf. Theory Appl. Workshop*, 2006, pp. 1–5.

[40] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[41] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, 2003, pp. 1–10.

[42] R. Aravind, M. Civanlar, and A. Reibman, "Packet loss resilience of MPEG-2 scalable video coding algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, pp. 426–435, Oct. 1996.

[43] R. Singh, A. Ortega, L. Perret, and W. Jiang, "Comparison of multiple description coding and layered coding based on network simulations," in *Proc. Dispersity Routing Store Forward Netw.*, 2000, pp. 929–939.

[44] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts: Process Scheduling*. Hoboken, NJ, USA: Wiley, 2010.

[45] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability benefit of network coding," Comput. Sci. Dept., Univ. Mass. Amherst, Amherst, MA, USA, Tech. Rep. 07-08, 2007.

[46] M. Jerrum and A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," in *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum ed. Boston, MA, USA: PWS, 1996.

[47] M. Abramowitz and I. A. Stegun, Eds., "Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables," in *The 9th Printing*. New York, NY, USA: Dover, 1972.

[48] Voice Over IP—Per Call Bandwidth Consumption. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html

[49] C. Wang, D. Koutsonikolas, Y. Hu, and N. Shroff, "FEC-based AP downlink transmission schemes for multiple flows: Combining the reliability and throughput enhancement of intra- and inter-flow coding," *Perform. Eval.*, vol. 68, no. 11, pp. 1118–1135, Nov. 2011.

AQ11

Tuan Tran (M'XX) received the B.S. degree from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2000; the M.S. degree from the Polytechnic University of Turin, Turin, Italy, in 2006; and the Ph.D. degree from Oregon State University, Corvallis, OR, USA, in 2010, all in computer engineering.

He is currently an Assistant Professor with the College of Computer and Information Technology, Sullivan University, Louisville, KY, USA. Prior to that, he was a Postdoctoral Scholar with Arizona State University, Tempe, AZ, USA, and the University of Louisville from 2010 to 2012. His research interests include computer networks and cybersecurity, stochastic system modeling, network and channel coding, wireless systems, and multimedia networking.

Dr. Tran received the Best Paper Runner-Up Award at the IEEE International Conference on Computer Communication Networks in 2010 and the Jack Neubauer Memorial Award for the Best Systems Paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2012.

Thinh Nguyen (M'04) received the B.S. degree from the University of Washington, Seattle, WA, USA, in 1995 and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 2003. AQ12

He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA. He is interested in all things stochastic, with applications to signal processing, distributed systems, wireless networks, network coding, and quantum walks.

Dr. Nguyen has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the IEEE TRANSACTIONS ON MULTIMEDIA.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please check if "For the sake of expository" is properly changed to "For the sake of exposition", to make the statement clear; otherwise, kindly provide the correction.
AQ2 = Please check if changes in this paragraph are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ3 = Please check if MC is properly introduced here; otherwise, kindly provide the correction.
AQ4 = Please check if "reflex" is properly changed to "reflect"; otherwise, kindly provide the correction.
AQ5 = Please check if changes in this sentence are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ6 = Please check if changes here are properly captured, to make the statement clear; otherwise, kindly provide the correction.
AQ7 = Please check if citation for Fig. 15(a) should be captured as citation for Fig. 15 only, to match the figures; otherwise, kindly provide the correction.
AQ8 = Note that reference [3] and [7] are the same. Therefore, reference [7] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ9 = Note that reference [5] and [39] are the same. Therefore, reference [39] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ10 = Note that reference [26] and [38] are the same. Therefore, reference [38] was deleted from the list. Citations were renumbered accordingly. Please check.
AQ11 = Please provide membership history of author "Tuan Tran."
AQ12 = Kindly provide the area of study for both B.S. and Ph.D. degrees.

END OF ALL QUERIES