A New Capacity-Approaching Protocol for General 1-to-K Broadcast Packet Erasure Channels with ACK/NACK

Chih-Hua Chang and Chih-Chun Wang, School of ECE, Purdue University

***Abstract*—The capacity region of 1-to-K broadcast packet erasure channels with ACK/NACK is known for some scenarios, e.g., $K \leq 3$, etc. However, existing achievability schemes either require knowing the target rate $\vec{R}$ in advance, and/or have a complicated description of the achievable rate region that is difficult prove whether it matches the capacity or not. This work proposes a new network coding protocol with the following unique set of features: (i) Its achievable rate region is identical to the capacity region for *all* the scenarios in which the capacity is known; (ii) Its achievable rate region is much more tractable that existing works and has been used to derive new capacity rate vectors; (iii) It employs *sequential encoding* that naturally handles dynamic packet arrivals; (iv) It automatically adapts to unknown packet arrival rates $\vec{R}$; (v) It is based on $\mathsf{GF}(q)$ with $q > K$. Numerically, for $K = 4$, it admits an average control overhead 2–4% (assuming each packet has 1000 bytes), average encoding memory usage 48.5 packets, and average per-packet delay 94.8 time slots, when operating at 95% of the capacity.**

I. Introduction

The broadcast channel (BC) is an important subject of network information theory. Recently, BC with causal channel feedback is considered along the lines of degree-of-freedom analysis [1] and the ACKnowledgement-feedback for broadcast packet erasure channels (PEC) [2]. The PEC setting is widely used to model potential packet loss in modern wireless communication networks [2]–[4].

Network coding is an important component when characterizing the capacity of broadcast PEC with causal ACK/NACK. For example, consider a transmitter s with two packets X and Y that need to be delivered to d_1 and d_2, respectively. If during previous transmissions d_1 overheard Y and d_2 overheard X, then s can send $[X + Y]$ that benefits both d_1 and d_2 simultaneously. This simple idea has been generalized for the 1-to-K broadcast PEC with causal ACK/NACK. The state-of-the-art results have characterized the capacity region for the cases of $K \leq 3$, symmetric channels, and when the *one-sided fairness* condition is satisfied. See [2]–[4].

For the scenarios in which the capacity region is still unknown, a simple outer bound is derived in [2], [3]. For the achievable rate region, a block-coding-based solution, called the *packet evolution* (PE) scheme, is proposed in [2], which attains the capacity outer bound for all the scenarios in which the capacity is known. The theoretic PE scheme nonetheless suffers several drawbacks and is difficult to implement in practice. *Drawback 1:* To achieve a given rate vector $\vec{R}$, PE has to first solve a linear-programming (LP) problem and uses the optimal LP solution to adjust the design parameters. However, in practice the packet arrival rates $\vec{R}$ are usually not known in advance. *Drawback 2:* The network coding opportunity exists *only after* some node d_k overhears some previous transmissions. Therefore, there is a strict causality relationship between when the coding opportunity is generated (through overhearing) and when one can combine packets to realize/consume the created coding opportunity. To rigorously quantify the achievable rate region, the PE scheme and the scheme in [3] relies on the knowledge of $\vec{R}$ to impose a strict "transmission order" that takes into account the causality constraint. Since a strict transmission order requires packets to "wait in the queues" for a long time before transmission, it incurs long queuing delay in practice.

Drawback 3: The theoretic PE scheme allows for asymptotically large memory, asymptotically large queuing/decoding delay, and asymptotically large complexity, which make it difficult to implement in practice.

The authors in [4] design a new sequential coding scheme that addresses the above drawbacks while using only binary XOR operations and admitting the desired feature of *instantaneous decodability*. In this work, we propose a new design that also addresses the above drawbacks but with the following unique combination of features. **Contribution 1:** We prove that its achievable rate region (ARR) is identical to the capacity region for *all* the scenarios in which the capacity is known. This is a strict improvement over [4], which only proves that their scheme achieves the capacity for symmetric BC with $K \leq 4$ destinations. **Contribution 2:** Our scheme has a systematic design, which results in a compact form of the ARR (only 2 equations). For comparison, the scheme in [4] reacts to various sub-cases differently, which makes the corresponding ARR harder to describe.[1] Also see our discussion in Section III-A. The tractability of our scheme allows us to discover new, previously unknown, optimal capacity vectors, see Corollary 2, which may not be possible with the ARR in [4] due to its complexity.

Contribution 3: Both this work and [4] employ sequential encoding rather than block coding; **Contribution 4:** Both this work and [4] are based on *back-pressure scheduling* and can

This work was supported in part by NSF grants ECCS-1407603, CCF-1422997, and CCF-1618475.

[1]The ARR in [4] is a concatenation of the virtual network description in its Section III and a flow-condition first introduced in [5], which takes several pages to carefully describe the necessary notations and expressions.

automatically adapt to any unknown $\vec{R}$ (without solving any LP problem). However, our back-pressure scheduler has a significantly smaller number of "competing actions" to consider during each back-pressure computation, which greatly reduces the complexity in practice. Take $K = 4$ for example, our scheme compares the back-pressures of only 15 different actions while [4] compares the back-pressures of 244 actions;
Contribution 5: Numerically, our scheme is very efficient in terms of the control overhead, encoding memory, and per-packet delay (queueing plus decoding delay), see Section IV. Compared to the scheme in [4], our scheme is based on $\mathsf{GF}(q)$ with $q > K$ and does not admit the useful feature of instantaneous decodability. On the other hand, the numerical results show that the total delay (queueing+decoding) is still quite manageable for practical scenarios.

II. Problem Formulation

For any positive integer K, we define $[K] \triangleq \{1, 2, \ldots, K\}$ and use the notation $2^{[K]} \triangleq \{S : S \subseteq [K]\}$ to represent the collection of all subsets of $[K]$.

We describe the capacity region following the traditional block-coding-based definition, even though our achievability scheme is a sequential encoder. Consider a 1-to-K broadcast PEC from source s to destinations d_k, $k \in [K]$. There are K independent packet streams, one stream for each d_k. For any block length n, we use $\mathbf{X}_k = \{X_{k,i} : i = 1, 2, \cdots, nR_k\}$, to denote the nR_k packets of stream k. The value of R_k is termed the rate for d_k. A packet $X_{k,i}$ is drawn randomly from a fixed finite field $\mathsf{GF}(q)$ with $q > K$.

At time slot $t = 1, 2, \cdots, n$, source s sends a packet $Y(t) \in \mathsf{GF}(q)$ as an input of the broadcast PEC. The channel outputs a K-dimensional vector $(Z_1(t), \cdots, Z_K(t))$ where for each k we have $Z_k(t) \in \{Y(t), *\}$. Herein $Z_k(t) = *$ means that the transmitted $Y(t)$ is "erased." We assume the 1-to-K broadcast PEC is *stationary* and *memoryless.* Let p_k denote the probability that d_k successfully receives the transmitted packet, i.e., $p_k \triangleq P(Z_k(t) = Y(t))$. We assume independence across destinations d_1 to d_K. For any disjoint subsets $S, T \in 2^{[K]}$, we define

$$p_{S\overline{T}} \triangleq \left(\prod_{k:k\in S} p_k\right)\left(\prod_{\tilde{k}:\tilde{k}\in T} (1 - p_{\tilde{k}})\right).$$

That is, $p_{S\overline{T}}$ denotes the probability that $Y(t)$ is received by all d_k in S but by none of the $d_{\tilde{k}}$ in T. We do not care whether $Y(t)$ is received by those destinations in $[K]\backslash(S \cup T)$. When S is empty, we use $p_{\overline{T}}$ as shorthand for $p_{\emptyset\overline{T}}$. For any time t, the *reception set* $S_{\text{rx}}(t)$ contains those k satisfying $Z_k(t) \neq *$.

For any time t, the source s sends a coded symbol

$$Y(t) = f_t((\mathbf{X}_1, \ldots, \mathbf{X}_K), [S_{\text{rx}}]_1^{t-1}),$$

which is a function $f_t(\cdot)$ that takes all information symbols $\{\mathbf{X}_k\}$ and the past reception sets $[S_{\text{rx}}]_1^{t-1} \triangleq \{S_{\text{rx}}(\tau) : \tau \in [t-1]\}$, where the inclusion of $[S_{\text{rx}}]_1^{t-1}$ models the causal ACK/NACK. In the end of time n, each d_k decodes

$$\hat{\mathbf{X}}_k = g_k([Z_k]_1^n, [S_{\text{rx}}]_1^n), \tag{1}$$

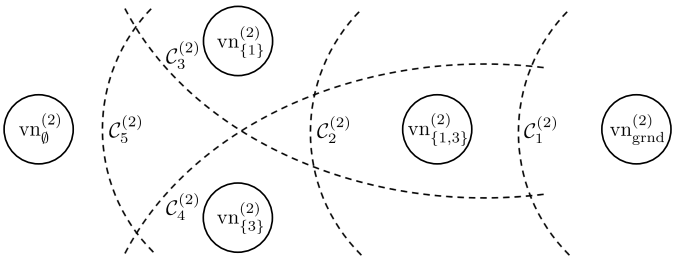

Fig. 1: The virtual nodes and proper-cuts for $k = 2$ and $K = 3$.

based on all the received packets $[Z_k]_1^n \triangleq \{Z_k(t) : t \in \{1, 2, \ldots, n\}\}$ and all the reception sets[2] $[S_{\text{rx}}]_1^n$.

A network code of length n is defined by the corresponding n encoding functions $f_t(\cdot)$ and K decoding functions $g_k(\cdot)$. The achievable rates of a 1-to-K broadcast PEC with ACK/NACK are then defined by

Definition 1. *Given any fixed* $\mathsf{GF}(q)$, $(R_1, \cdots, R_K)$ *is achievable if for any* $\epsilon \geq 0$, *there exists a network code of length* n *such that* $\Pr(\{\hat{\mathbf{X}}_k \neq \mathbf{X}_k\}) \leq \epsilon$. *The capacity region is the closure of all achievable rate vectors* $(R_1, \ldots, R_K)$.

Since for general channel parameters p_1 to p_K the exact capacity region is still unknown, we use the follow capacity outer bound as a benchmark.

A K-permutation (or simply *permutation*) is a bijective function from the set $[K]$ to itself $\pi : [K] \mapsto [K]$. Given any permutation π, for all $j \in K$ we define $S_j^{\pi} \triangleq \{\pi(l) : l \in [j]\}$ as the set of the first j elements according to π.

Proposition 1 ([2], [3])**.** *Any achievable rate* $(R_1, \ldots, R_K)$ *must satisfy the following* $K!$ *inequalities:*

$$\sum_{j=1}^{K} \frac{R_{\pi(j)}}{1 - p_{\overline{S_j^{\pi}}}} \leq 1, \quad \forall \pi. \tag{2}$$

III. Main Results

Our main result is a new achievability scheme and its achievable rate region. The main innovations include both the design of the new scheme and its performance analysis.

A. The Achievable Rate Region

Note that although the achievable rates are described as an LP problem, the actual execution of our protocol does not involve solving the LP. Instead, the performance of our scheme will automatically converge to the optimal LP solution.

Definition 2. *For any fixed* $k \in [K]$, *we define* 2^{K-1} virtual nodes *and label each of them by a subset* $S \subseteq [K]\backslash\{k\}$. *It is thus convenient to simply denote each virtual node as* $\mathrm{vn}_S^{(k)}$ *for all* $S \in 2^{[K]\backslash\{k\}}$. *We say a subset of* $\{\mathrm{vn}_S^{(k)} : \forall S\}$, *denoted by* $\mathcal{C}^{(k)}$, *is a* proper-cut *if the following conditions hold: (i)*

[2]When discussing whether a scheme can achieve the capacity, we assume $[S_{\text{rx}}]_1^n$ is known by d_k for free, as described in (1). When discussing a practical implementation, we term the cost of d_k not knowing $[S_{\text{rx}}]_1^n$ as the *control overhead*. See [2] for further discussion on control overhead.

$\mathrm{vn}_{\emptyset}^{(k)} \notin \mathcal{C}^{(k)}$*; (ii) If* $\mathrm{vn}_{S_1}^{(k)} \in \mathcal{C}^{(k)}$ *for some* S_1*, then for all* $S_2 \supseteq S_1$ *we must have* $\mathrm{vn}_{S_2}^{(k)} \in \mathcal{C}^{(k)}$.

An illustration is provided in Fig. 1 for $k = 2$ and $K = 3$, for which there are exactly 5 proper-cuts: $\mathcal{C}_1^{(2)} = \emptyset$, $\mathcal{C}_2^{(2)} = \{\mathrm{vn}_{\{1,3\}}^{(2)}\}$, $\mathcal{C}_3^{(2)} = \{\mathrm{vn}_{\{1\}}^{(2)}, \mathrm{vn}_{\{1,3\}}^{(2)}\}$, $\mathcal{C}_4^{(2)} = \{\mathrm{vn}_{\{3\}}^{(2)}, \mathrm{vn}_{\{1,3\}}^{(2)}\}$, and $\mathcal{C}_5^{(2)} = \{\mathrm{vn}_{\{1\}}^{(2)}, \mathrm{vn}_{\{3\}}^{(2)}, \mathrm{vn}_{\{1,3\}}^{(2)}\}$. Note that there is a "ground node" $\mathrm{vn}_{\mathrm{grnd}}^{(2)}$ being added in Fig. 1. One can see that the node set $\mathcal{C}_i^{(i)} \cup \left\{\mathrm{vn}_{\mathrm{grnd}}^{(2)}\right\}$ is indeed a *cut* separating $\mathrm{vn}_{\emptyset}^{(2)}$ from $\mathrm{vn}_{\mathrm{grnd}}^{(2)}$, which thus gives the name "proper-cut."

Proposition 2. $\vec{R} \triangleq (R_1, \ldots, R_K)$ *can be achieved by our proposed scheme if there exists* $\left\{x_T \geq 0 : \forall T \in 2^{[K]} \backslash \{\emptyset\}\right\}$ *satisfying [Condition 1:] The time-sharing condition:*

$$\sum_{T: T \in 2^{[K]} \backslash \{\emptyset\}} x_T \leq 1; \tag{3}$$

and [Condition 2:] The min-cut condition: For all k *and for all* $\mathcal{C}^{(k)}$ *being a proper cut,*

$$\sum_{\substack{S: S \in 2^{[K]\backslash\{k\}}, \\ \mathrm{vn}_S^{(k)} \notin \mathcal{C}^{(k)}}} x_{S \cup \{k\}} \cdot \left(p_k + \sum_{\substack{\forall S_X: \\ S_X \in \mathcal{F}(k,S,\mathcal{C}^{(k)})}} p_{S_X \overline{[K]\backslash(S_X \cup S)}} \right) \geq R_k, \tag{4}$$

where the set $\mathcal{F}(k, S, \mathcal{C}^{(k)})$ *is defined by*

$$\mathcal{F}(k, S, \mathcal{C}^{(k)}) \triangleq \Big\{ S_X \in 2^{[K]\backslash(S\cup\{k\})} : \exists \tilde{S} \subseteq (S_X \cup S) \text{ s.t. } \mathrm{vn}_{\tilde{S}}^{(k)} \in \mathcal{C}^{(k)} \Big\}.$$

Corollary 1. *For all the scenarios in which the capacity is known, i.e., symmetric channels,* $K \leq 3$*, one-sided fairness, see [2], the above achievable rate region matches the capacity.*

Actually, the achievable rate region in Proposition 2 meets the outer bound in many other scenarios not specified in the above corollary. For example,[3] we also have

Corollary 2. *For* $K = 4$ *and* $\vec{p} = (p_1, p_2, p_3, p_4) = (\frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{4}{7})$*, the rate vector* $\vec{R} = (R_1, R_2, R_3, R_4) = (\frac{96}{1193}, \frac{672}{5965}, \frac{288}{1193}, \frac{5856}{25053})$ *does not belong to any of the scenarios in Corollary 1. However, it is in the boundary of the outer bound* (2) *and also lies within the achievable rate region in Proposition 2. Therefore, it is an optimal capacity vector for the given* $\vec{p}$*.*

The proofs of Proposition 2 and Corollaries 1 and 2 are omitted due to the space constraint.

We now explain the intuition of Proposition 2. As illustrated in Fig. 1, for each k we have a virtual network of $2^{K-1} + 1$ nodes, i.e., $\{\mathrm{vn}_S^{(k)} : S \in 2^{[K]\backslash\{k\}}\}$ and $\mathrm{vn}_{\mathrm{grnd}}^{(k)}$. We then connect these $2^{K-1} + 1$ nodes in the following way. (See Fig. 2 for illustration.) For each $\mathrm{vn}_S^{(k)}$, we add (i) an edge of capacity p_k that ends in $\mathrm{vn}_{\mathrm{grnd}}^{(k)}$; (ii) $2^{K-(|S|+1)}$ auxiliary nodes, each denoted by a subset $S_X \in 2^{[K]\backslash(S\cup\{k\})}$. For example if $K = 3$ and we focus on $\mathrm{vn}_{\{1\}}^{(2)}$, then we add two auxiliary nodes denoted by $S_X = \emptyset$ and $S_X = \{3\}$, respectively. The edge connecting $\mathrm{vn}_S^{(k)}$ and its auxiliary node S_X have capacity $p_{S_X \overline{[K]\backslash(S\cup S_X)}}$. Finally, for each auxiliary node S_X of $\mathrm{vn}_S^{(k)}$, we add an edge of infinity capacity to virtual node $\mathrm{vn}_{\tilde{S}}^{(k)}$ for all $\tilde{S} \subseteq (S \cup S_X)$. For example, if $K = 3$ and we focus on $\mathrm{vn}_{\{1\}}^{(2)}$, then we add 2 infinite-capacity edges from $S_X = \emptyset$ to $\mathrm{vn}_{\emptyset}^{(2)}$ and $\mathrm{vn}_{\{1\}}^{(2)}$, respectively. We then place a source src_k at $\mathrm{vn}_{\emptyset}^{(k)}$ and a sink snk_k at $\mathrm{vn}_{\mathrm{grnd}}^{(k)}$. Once the K virtual networks is determined (one for each k), we refer them all together as a big virtual network with K sub-networks.

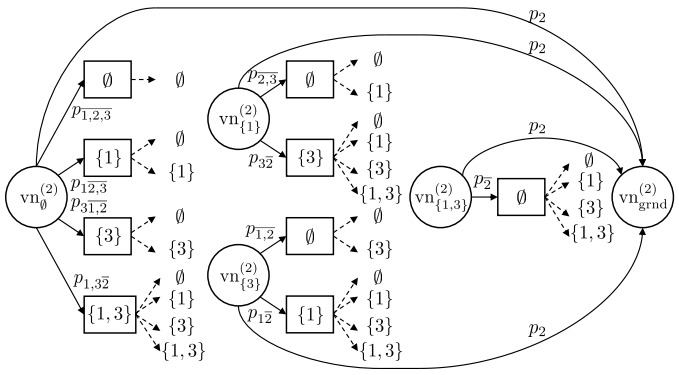

Fig. 2: The virtual sub-network for $k = 2$ and $K = 3$, where the virtual nodes are represented by circles and the auxiliary nodes by squares. The dotted edge marked by a set S is actually connected to $\mathrm{vn}_S^{(2)}$ with ∞ capacity.

We now elaborate the relationship between the virtual network and the network coding operations. For any $T \in 2^{[K]}$, if for all $k \in T$ there exists a flow-k packet X_{k,i_k} that has previously been overheard by all other destinations $d_{\tilde{k}}$ for all $\tilde{k} \in T\backslash\{k\}$, then we can send a linear sum $\sum_{k\in T} X_{k,i_k}$ that benefits all d_k, $k \in T$, simultaneously. Roughly speaking, the value of x_T in (3) corresponds to the frequency that we send a linear sum of $|T|$ packets, one packet from each flow $k \in T$. Since each time slot can only uses one specific T, the frequencies $\{x_T\}$ must satisfy the time-sharing condition.

Intuitively, we then let $\mathrm{vn}_S^{(k)}$ represent the flow-k packets that have been overheard by all $d_{\tilde{k}}, \tilde{k} \in S$. Since combining flows of $k \in T$ can simultaneously benefit all flows $k \in T$, a larger x_T value should help move more packets out of $\mathrm{vn}_S^{(k)}$ where $S = T\backslash\{k\}$. That is why we scale the virtual network edge capacity in (4) by $x_{S\cup\{k\}}$ all k and all $S \in 2^{[K]\backslash\{k\}}$. The min-cut condition in (4) then implies that the min-cut value for the k-th sub virtual network is no less than R_k, the target rate. By the min-cut/max-flow theorem, if the min-cut is no less than R_k, then we can also find the max-flow that achieves R_k in the virtual sub-network. This can then be translated back

[3]Corollary 2 is not unique in any sense. In fact, for $K = 4$ we have run 10^5 numerical trials with different $\vec{p}$ and we have not found any $\vec{R}$ that is in the outer bound but not in Proposition 2.

to the achievability of network codes.

Our virtual network representation and the corresponding achievable rate characterization is significantly different from [4]. Firstly, the virtual network in [4] uses 1-to-1 edges, i.e., from $\mathrm{vn}_S^{(k)}$ to another $\mathrm{vn}_{\tilde{S}}^{(k)}$. In contrast, by letting $\mathrm{vn}_S^{(k)}$ first connect to an auxiliary node labeled by S_X and then adding edges of infinity capacity from S_X to many virtual nodes $\mathrm{vn}_{\tilde{S}}^{(k)}$ for all $\tilde{S} \subseteq (S \cup S_X)$, we essentially create a 1-to-many *hyper-edge* from $\mathrm{vn}_S^{(k)}$ to $\{\mathrm{vn}_{\tilde{S}}^{(k)} : \tilde{S} \subseteq (S \cup S_X)\}$, see Fig. 2; Secondly, the virtual network in [4] is acyclic as the packets only move to queues of *higher levels*. Our construction contains many cycles.

The new cyclic, hyper-graph-based virtual network is the key to a compact and tractable form of the achievable rate region (ARR). This is an important feature that is absent in [4]. To illustrate the difference in terms of tractability, for $K = 4$ and asymmetric channels, the LP problem of our ARR has only 15 variables with a simple (regular) form of cut-based conditions. This makes it straightforward to prove additional results like Corollaries 1 and 2. On the other hand, the ARR of the most general *inter-level scheme* in [4] has 244 variables with several different ways of specifying the coefficients of the linear inequalies, which makes it more difficult to use. Even the less powerful *intra-level scheme*, which is not capacity-achieving for the general $K = 3$ case, the results in [4] still involve 112 variables.

B. A New Sequential Network Coding Protocol

We now describe our scheme, which does not need to solve any LP (even though its performance is described by LP).

Component 1: The queues at source s. There are $K \cdot 2^{K-1}$ queues denoted by $Q_S^{(k)}$ for all $k \in [K]$ and $S \in 2^{[K]\setminus\{k\}}$. Each entry in the queue contains three parts: The header length L, the header content (or just the *header*), and the payload. The header contains a list of L tuples: (k_1, i_1, c_1) to (k_L, i_L, c_L); The payload part contains the linear sum $\sum_{l=1}^{L} c_l \cdot X_{k_l,i_l}$.

Namely, the payload is the linear sum of the i_l-th packet of the k_l-th flow, multiplied by a coefficient c_l, for all $l \in [L]$. The header length L and the header can be viewed as a compressed version of the *global encoding kernel* of the payload, see [6].

Component 2: Sequential packet arrivals. Whenever the i-th flow-k packet arrives at s, denoted by $X_{k,i}$, we store it in the queue $Q_\emptyset^{(k)}$ with $L = 1$, the header containing the tuple $(k, i, 1)$, and the payload being the just arrived packet.

Component 3: Selecting the set $T^* \in 2^{[K]}$ of flows to be added together. For each time t, we select a new T^* as follows. For all $k \in [K]$ and all $\check{S} \in 2^{[K]\setminus\{k\}}$, we compute

$$q(k, \check{S}) \triangleq \min_{\tilde{S} \in 2^{\check{S}}} |Q_{\tilde{S}}^{(k)}|, \tag{5}$$

where $|Q_{\tilde{S}}^{(k)}|$ is the number of entries in $Q_{\tilde{S}}^{(k)}$. The minimum operation taken in (5) is specifically designed to mimic the *hyper-edge*-based virtual network construction discussed in Section III-A. After computing $q(k, \check{S})$, we then compute the *back-pressure* for queue $Q_S^{(k)}$ by

$$\mathrm{bp}(Q_S^{(k)}) \triangleq |Q_S^{(k)}| - \sum_{S_X \in 2^{[K]\setminus(S\cup\{k\})}} p_{S_X \overline{[K]\setminus(S\cup S_X)}} q(k, S \cup S_X). \tag{6}$$

The target set T^* is then selected as follows.

$$T^* = \arg\max_{\text{non-empty } T \in 2^{[K]}} \sum_{k \in T} \mathrm{bp}(Q_{T\setminus\{k\}}^{(k)}). \tag{7}$$

That is, since the packets in queues $\{Q_{T\setminus\{k\})}^{(k)} : k \in T\}$ will be added together, their individual back pressures are also summed together. We thus choose the T^* with the largest sum.

Component 4: Encoding. The payload of the to-be-transmitted packet is set to

$$\mathsf{PL}_{\text{new}} = \sum_{k \in T^*} \beta_k \cdot \mathsf{PL}_{\text{old},k},$$

where $\mathsf{PL}_{\text{old},k}$ is the payload of the head-of-line entry[4] of $Q_{T^*\setminus\{k\}}^{(k)}$ for all $k \in T^*$. That is, the new payload is the linear sum of the existing payloads multiplied by β_k. For ease of exposition, one can assume β_k is chosen uniformly randomly from a sufficiently large $\mathsf{GF}(q)$. The list of tuples in the new header, denoted by H_{new}, would then be the compressed *global encoding kernel* of the new payload. Source s then transmit $(\mathsf{H}_{\text{new}}, \mathsf{PL}_{\text{new}})$ as the coded packet.

Component 5: Packet movement between the queues. This component describes how each entry of $Q_S^{(k)}$ moves after s receives ACK/NACK from $\{d_k\}$. We describe the packet movement for one specific k_0 while the same mechanism has to be applied to all $k \in [K]$.

If $k_0 \notin T^*$, then no entry in $\{Q_S^{(k_0)} : \forall S\}$ will move. If $k_0 \in T^*$, then we consider one and only one queue: $Q_{T^*\setminus\{k_0\}}^{(k_0)}$. We first remove[5] the head-of-line entry of $Q_{T^*\setminus\{k_0\}}^{(k_0)}$. If d_{k_0} received the coded transmission, then no further action is needed. If d_{k_0} did not receive the coded transmission, we will inject the pair $(\mathsf{H}_{\text{new}}, \mathsf{PL}_{\text{new}})$ to another queue.

The new destination queue is decided as follows. Define $\check{S} = (T^*\setminus\{k_0\}) \cup S_{\text{rx}}$ where the reception set S_{rx} is computed from ACK/NACK. Define $\tilde{S}^*$ as the $\tilde{S}$ that minimizes the expression $q(k_0, \check{S})$ in (5). Then we choose $Q_{\tilde{S}^*}^{(k_0)}$ as the destination queue.

Component 5 is the most distinct feature of our protocol. Unlike existing solutions that predetermine where the packets in the virtual network will move (usually in an acyclic fashion), Component 5 decides the movement dynamically with the help of queue lengths, possibly in a cyclic way. For example, consider $K = 3$ and suppose that at time t, the target set is $T^* = \{1, 2, 3\}$. I.e., packets in three different queues (one for each destination d_k) are used to generate the new packet $(\mathsf{H}_{\text{new}}, \mathsf{PL}_{\text{new}})$ for transmission. Also suppose after

[4]In the case that a queue $Q_{T^*\setminus\{k\}}^{(k)}$ is empty, we assume its head-of-line entry has payload being all 0.

[5]In the degenerate case in which $Q_{T^*\setminus\{k_0\}}^{(k_0)} = \emptyset$, then no further action will be needed (no entry removal, no new entry injection, etc).

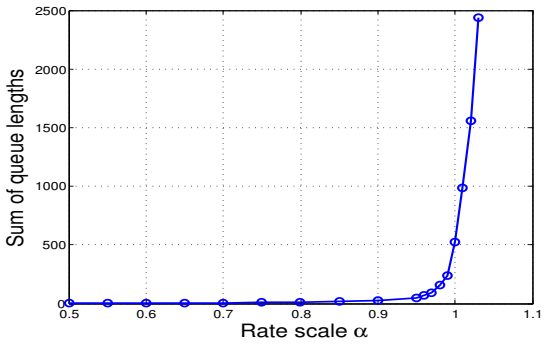

Fig. 3: The sum of queue lengths $\sum_{k,S} |Q_S^{(k)}|$ for different α.

transmission, the reception set is $S_{\text{rx}} = \{3\}$. The remaining question is how the packets will move within the queues.

Consider the packets for d_2, i.e., $k_0 = 2$. Suppose the queue lengths for d_2 are $|Q_{\{\emptyset\}}^{(2)}| = 5$, $|Q_{\{1\}}^{(2)}| = 3$, $|Q_{\{3\}}^{(2)}| = 1$, and $|Q_{\{1,3\}}^{(2)}| = 2$. By Component 5, the head-of-line packet in $Q_{T^*\backslash\{k_0\}}^{(k_0)} = Q_{\{1,3\}}^{(2)}$ will be removed. Since $\check{S} = (T^*\backslash\{k_0\}) \cup S_{\text{rx}} = \{1,3\}$ we compute

$$q(2,\{1,3\}) = \min\left\{|Q_{\{\emptyset\}}^{(2)}|, |Q_{\{1\}}^{(2)}|, |Q_{\{3\}}^{(2)}|, |Q_{\{1,3\}}^{(2)}|\right\} = 1$$

and the minimizing $\tilde{S}^* = \{3\}$. The new packet $(\mathsf{H}_{\text{new}}, \mathsf{PL}_{\text{new}})$ is then injected to queue $Q_{\tilde{S}^*}^{(k_0)} = Q_{\{3\}}^{(2)}$.

The above 5 components completely describe our scheme. The main difference to [4], which also uses back-pressure schedulers, is as follows. Enabled by a new hyper-edge-based virtual network, our scheme has a significantly smaller number of competing actions, i.e., each time slot we only need to choose T^* out of $2^K - 1$ different non-empty $T \in 2^{[K]}$. For comparison, the scheme in [4] compares 244 number of competing actions for $K = 4$ and significantly more[6] for general K. One can see that our construction not only has a more tractable achievable rate region but its actual implementation is simpler with straightforward choice of T^*.

The achievability of our scheme is defined by the following propositions. We first define $Q_S^{(k)}(t)$ as the queue at time t.

Proposition 3. *For any fixed time t and any fixed $k \in [K]$, if we deliver to d_k all the entries that are in the queues $\{Q_S^{(k)}(t) : S \in 2^{[K]\backslash\{k\}}\}$ through a separate noiseless channel, then d_k can decode all $X_{k,i}$ that have already arrived at s.*

Proposition 3 shows that the queues at s indeed corresponds to the packets that have not been "delivered" to $\{d_k\}$.

Proposition 4. *If the arrival of flow-k packets is Bernoulli distributed with parameter R_k and if $\vec{R} = (R_1, \cdots, R_K)$ is in the interior of the rate region described in Proposition 2, then for all $k \in [K]$ and $S \in 2^{[K]\backslash\{k\}}$*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=1}^{t} E\left\{|Q_S^{(k)}(\tau)|\right\} < \infty.$$

Jointly Propositions 3 and 4 show that even when $t \to \infty$, the number of yet-to-be-delivered packets are bounded. The rate vector $\vec{R}$ is thus achievable.

IV. Simulation Results

In this section, we assume that the arrival of the flow-k packets is i.i.d. Bernoulli with success probability R_k. Assume $K = 4$ and $(p_1, p_2, p_3, p_4) = (\frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{4}{7})$. Corollary 2 shows that $\vec{R}^* \triangleq (R_1^*, R_2^*, R_3^*, R_4^*) = (\frac{96}{1193}, \frac{672}{5965}, \frac{288}{1193}, \frac{5856}{25053})$ is the optimal capacity. We then set the actual arrival probability to be $\vec{R}_\alpha = (R_1, R_2, R_3, R_4) = \alpha\vec{R}^*$ where $0 < \alpha \leq 1$ measures how closely we are operating at the capacity. Each simulation trial lasts for 10^5 time slots.

To measure the sum of queue lengths $\sum_{k,S} |Q_S^{(k)}|$, we run 10 trials and Fig. 3 reports the average for different α values. For any $\alpha < 1$, the sum of queue lengths remains very small. Note that the queue lengths correspond to how many packets s needs to store in the memory during sequential encoding, which is around 48.5 packets when $\alpha = 0.95$.

The control overhead is measured by the length of the header of each transmission and the per-packet totaly delay is measured by the time span between a packet first arriving at source s until it is fully decoded at destination d_k (queueing plus decoding delay). We run 5 trials with $\alpha = 0.95$. The average header length and average per-packet total delay are 10.5 and 94.8, respectively. If each (k, i, c) tuple in the header takes 4 bytes, then in average 42 bytes are used for headers, 4.2% of a regular packet of 1000 bytes.

V. Conclusion

We have presented a new network coding protocol for 1-to-K broadcast packet erasure channels with causal ACK/NACK. The achievable rate region matches the capacity region for all the scenarios in which the capacity is known. The proposed scheme has many desirable features, including sequential encoding and being adaptive to unknown arrival rates.

References

[1] M. A. Maddah-Ali and D. Tse, "Completely stale transmitter channel state information is still very useful," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4418–4431, Jul. 2012.

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

[3] M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Multiuser broadcast erasure channel with feedback–capacity and algorithms," *IEEE Trans. Inf. Theory*, vol. 59, no. 9, pp. 5779–5804, Sep. 2013.

[4] S. Athanasiadou, M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Stable XOR-based policies for the broadcast erasure channel with feedback," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 476–491, Feb. 2016.

[5] G. S. Paschos, L. Georgiadis, and L. Tassiulas, "Scheduling with pairwise XORing of packets under statistical overhearing information and feedback," *Queueing Syst.*, vol. 72, no. 3, pp. 361–395, 2012.

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

[6]The exact number of competing actions in [4] for general K is hard to estimate, which involves finding all possible coding choices that are compatible to the *Basic Coding Rule*. A loose lower bound on the number of competing actions is $B_{K+1}-2$, where B_{K+1} is the $(K+1)$-th Bell number. For example, $B_6 = 203$, $B_9 = 21147$, $B_{11} = 678570$, $B_{16} \approx 10^{10}$, and $B_{19} \approx 5.8 \times 10^{12}$.