Sending Perishable Information: Coding Improves Delay-Constrained Throughput Even for Single Unicast

Chih-Chun Wang, Senior Member, IEEE, and Minghua Chen, Senior Member, IEEE

Abstract—This work considers network communications under a hard timeliness constraint, where a source node streams perishable information to a destination node over a directed acyclic graph subject to a hard delay constraint. Transmission along any edge incurs unit delay, and it is required that every information bit generated at the source at the beginning of time $t$ to be received and recovered by the destination at the end of time $t + D - 1$ where $D > 0$ is the maximum allowed end-to-end delay. We study the corresponding delay-constrained unicast capacity problem.

This work presents the first example showing that network coding (NC) can achieve strictly higher delay-constrained throughput than routing even for the single unicast setting and the NC gain can be arbitrarily close to 2 in some instances. This is in sharp contrast to the delay-unconstrained ($D = \infty$) single-unicast case where the classic min-cut/max-flow theorem implies that coding cannot improve throughput over routing. Motivated by the above findings, a series of investigation on the delay-constrained capacity problem is also made, including (i) an equivalent multiple-unicast representation based on a time-expanded graph approach, (ii) a new delay-constrained capacity upper bound and its connections to the existing routing-based results [Ying, et al. 2011], (iii) an example showing that the penalty of using random linear NC can be unbounded, and (iv) a counter example of the tree-packing Edmonds’ theorem in the new delay-constrained setting. Built upon the time-expanded graph approach, we also discuss how our results can be readily extended to cyclic networks. Overall, our results suggest that delay-constrained communication is fundamentally different from the well-understood delay-unconstrained one and call for investigation participation.

Index Terms—Network coding; max-flow/min-cut; delay-constrained communications; single unicast; network information theory.

I. INTRODUCTION

Consider a network modeled as a directed acyclic graph $G$, for which each edge has a capacity constraint and incurs a unit transmission delay. A link with long delay is modeled as multiple edges in tandem, each with unit delay. Except when specified otherwise, we consider exclusively a delay-constrained single-unicast scenario where a single source node, denoted as $s$, streams perishable information to a single destination node, denoted as $d$, over the graph $G$. Every information bit generated at $s$ at the beginning of time $t$ has to be received and recovered by $d$ by the end of time $t + D - 1$. Namely, the maximum allowed end-to-end communication delay of any packet is $(t + D - 1) - t + 1 = D$, where the value of $D$ is specified by the delay requirement of the applications. For easier reference, we use “at time $t$” to refer to “at the beginning of time $t$”. Since the end of time $t + D - 1$ is equivalent to the beginning of time $t + D$, our setting simply means that each packet generated at time $t$ needs to be decoded at time $t + D$.

In this paper, we formally define and study the delay-constrained unicast capacity problem, which characterizes the maximum rate at which $s$ can stream perishable information to $d$ subject to the delay constraint $D$.

The problem is important for delay-sensitive multimedia communication systems, and for delivering real-time control messages for cyber-physical systems. For example, in video conferencing, the system designer may want to maximize video throughput (thus video quality) subject to a video-delivery delay constraint of 250ms, to ensure interactive conferencing experience [3]. Similarly, in cyber-physical systems, time-critical applications impose latency constraints within which data or control messages must reach their targeting entities [2]. In general, an optimal delay-constrained communication scheme needs to decide the optimal routes of the information flow in space in order to fully utilize all the link capacity resources, while simultaneously tracking the delay of individual packets in time to ensure the packets can arrive at $d$ and the information can be recovered before expiration. The design problem becomes even more involved when we move away from the traditional routing paradigm (also known as the store-and-forward paradigm) and allow for network coding (NC) [1], [31] at intermediate nodes that intelligently mix the information content in the packets before forwarding them. Such a 3-way coupling among space, time, and NC choices creates a unique challenge and our understanding of delay-constrained network capacity is still nascent. In the following, we briefly illustrate some unique phenomena of the delay-constrained setting.

When $D$ is sufficiently large (e.g., larger than the end-to-end delay of the longest path between $s$ and $d$), any communication...
scheme can always meet the delay constraint. In this case, the delay-unconstrained (since $D = \infty$) single-unicast capacity can be characterized by the classic max-flow/min-cut theorem, and an optimal routing solution can be obtained in polynomial time using the Edmonds-Karp algorithm [7] (an improvement of the Ford-Fulkerson algorithm [11]), the push-&-relabel algorithm [13], simple linear programming (LP) solvers [42], [44], and a network-coding-based sounding approach [36].

See [12] for further references. Since optimal routing already achieves the capacity, i.e., the min-cut value, NC cannot improve throughput over optimal routing when there is only one unicast flow in the network.

The story changes completely when $D$ is small (i.e., when the delay constraint is active). For example, the delay-constrained unicast routing capacity has to be computed by the concept of soft edge-cuts [24], [45], which is different from the standard graph-theoretic notion of edge cuts. Also, as will be illustrated in Section II-C, there are some simple network instances for which optimal routing can achieve strictly higher delay-constrained throughput than random linear network coding (RLNC), a sharp contrast to the delay-unconstrained case in which both RLNC and optimal routing can achieve the single-unicast capacity [18].

Overall, we observe that the landscape of delay-constrained unicast is fundamentally different from the well-understood delay-unconstrained one. In this paper, we study the delay-constrained unicast capacity problem and make the following contributions:

Firstly, the single-unicast delay-constrained network capacity problem is officially formulated, which allows for rigorous future investigation. One immediate result of the delay-constrained single-unicast formulation is an equivalent delay-unconstrained multiple-unicast time-expanded network representation.

Secondly, this work shows for the first time in the literature that for delay-constrained traffic, NC can achieve strictly higher throughput than optimal routing even for single unicast and the NC gain\(^1\) can be arbitrarily close to 2.

Such a result is interesting in the following sense. Most of the Internet traffic is unicast. One of the fundamental results in NC is that routing achieves the single-unicast capacity when there is no delay constraint. This implies that to capitalize the NC benefits for delay-insensitive unicast traffic, one has to perform NC over multiple coexisting flows, the so-called inter-flow NC. It is known that designing the optimal inter-flow NC scheme is a notoriously hard problem, see [4], [16], [20], [22], [26]–[28], [32], [37]–[41] and the references therein. What exacerbates the problem is that even if we can design an optimal inter-flow NC scheme in a theoretic setting, in practice inter-flow NC requires additional coordinations among participating flows, including the tasks of hand-shaking, synchronization, joint buffer management, etc. Our result suggests that one may use NC to improve the performance of delay-sensitive traffic over optimal routing without any coordination among coexisting network flows.

\(^1\)Intuitively, the NC gain of a given network instance is defined as the ratio between the optimal NC capacity and the optimal routing rate. A formal definition is given later in (12).

Thirdly, based on a time-expanded graph approach, we propose a new upper bound on the delay-constrained unicast NC capacity, which provides deeper understanding to the overall delay-constrained network communication problem and sheds further insights to the existing routing-based delay-constrained results in [45] through the concept of integrality gap.

Fourthly, various aspects of the delay-constrained capacity are investigated, including the potentially unbounded penalty\(^2\) of RLNC over routing; and a revisit of the Edmonds’ tree packing results [6] for the delay-constrained setting.

This paper is organized as follows. Section II compares this work to the existing work on network-coding-based delay minimization, and summarizes the existing results on routing (store-and-forward) based solutions. Section III formulates the delay-constrained capacity problem. Section IV states the main results of this work, including (i) An equivalent time-expanded multiple unicast formulation; (ii) the NC gain over routing; and (iii) A new delay-constrained capacity upper bound. Section V contains various other results on delay-constrained capacity, including (iv) The unboundedness of the penalty of using RLNC; and (v) An example showing that Edmonds’ tree packing theorem no longer holds for the delay-constrained setting. Some remarks on generalizing the above results from acyclic to cyclic networks are also provided in Section V. Section VI concludes this work. In this paper, we formally present our discovery in the main body and leave the majority of the proofs in the appendices so that the logic flow of the statements is uninterrupted.

II. COMPARISON TO THE EXISTING WORK

A. Existing results on delay-related network coding

1) Delay minimization of network coding: Decoding delay of NC is a very well studied problem, see [5], [9], [10] and the references therein. Most of the existing works along this line focus on how to minimize the decoding delay when using NC to attain the best possible throughput for delay-insensitive traffic. Namely, attaining the absolute optimal throughput is of the highest priority, and minimizing the delay is to ensure that the extra (decoding) delay incurred by NC is not excessive. Many of the existing results also focus on 1-hop networks with random packet erasure.

In contrast with the existing throughput-centric delay-minimization studies [5], [10], this work is delay-centric. Namely, we consider a hard delay constraint such that any packets that experience delay longer than $D$ time slots are deemed useless and discarded. With the highest priority being the delay constraint, we focus on maximizing the throughput over any given error-free multi-hop network. This approach is a significant departure from the existing works on minimizing NC decoding delay.

In a way similar to this work, [9] studies the delay-capacity tradeoff. On the other hand, [9] focuses exclusively on the single-multicast scenario, assumes there is no link

\(^2\)Intuitively, the RLNC penalty of a network instance is defined as the ratio between the optimal routing rate and the optimal RLNC rate. A formal definition is given later in (27).
propagation delay, and studies the decoding delay related to the (large) underlying finite field size. For comparison, this work studies the single-unicast setting and focuses exclusively on the propagation delay of each link.

2) Delay-aware network code designs: Another line of network coding studies on delay is the delay-aware network code design, e.g., [8], [14], [25], [29], [35] and the references therein. In general, the goal along this line of work is to develop optimal network codes that maximize the throughput (achieving the min-cut value) while assuming that packets that arrive with long delay is as useful as packets with short delay. The difference is that the optimal code design now has to take into account the delay incurred in each edge/link in a way related to the traditional convolutional codes. The ring of rational power series is often used as the algebraic foundation of the corresponding network code design and study.

Among them, [14] studies the delay-constrained decodability problem, an identical setting as of this work. Specifically, [14] answers the question that for any given linear network code (with all the local encoding kernels explicitly specified), what is the necessary and sufficient algebraic condition that determines whether a given network code can sustain a delay-constrained throughput $R$. Therefore, for any given linear network code, one can use the results in [14] to verify whether it can achieve the targeted delay-constrained rate. For completeness, we will restate this important algebraic result [14] in Appendix I.

For comparison, instead of passively verifying the delay-constrained performance of any given linear network code, our work studies the best possible linear/non-linear network codes for a given delay constraint and compares it to that of the existing routing-based solutions, see Section II-B. We are particularly interested in exploring the graph-theoretic and information-theoretic connections, which is different from the algebraic approach in [14].

3) Time-stamped communications with infinite backlog: There also exist some works on infinite-backlog but timely delivery [32], [33]. In those settings, all the packets are available before the transmission starts and each packet has a fixed expiration time with uniform spacing. In terms of practical applications, our setting focuses more on the video conferencing scenarios where each message packet, once generated, needs to be delivered within a delay constraint. For comparison, the results in [32], [33] are more related to on-demand movie playback where all packets have already been stored in the data center and the design goal is to ensure smooth sequential packet delivery (thus the expiration time with uniform spacing). The main difference is that the expiration time of each packet is now predetermined and does not depend on when the packet is injected into the network.

For example, under the settings of [32], [33] it is a viable strategy to buffer the packets in certain ways to ensure smooth playback. In contrast, buffering packets is a very poor strategy in our delay-constrained setting since buffering packets will significantly increase the end-to-end delay and thus decrease the delay-constrained throughput.

B. Existing and some new results when only store- & forward is allowed

We model the network as a finite directed acyclic graph $G = (V, E)$, where $V$ is the node set and $E$ is the edge set. We use $\text{In}(v)$ and $\text{Out}(v)$ to denote the collections of the incoming and outgoing edges of $v$, respectively. For any $e = (u, v) \in E$, we define $\text{tail}(e) \triangleq u$ and $\text{head}(e) \triangleq v$. Each edge has a capacity constraint $c_e$ and incurs unit delay. Links with long delay are thus modeled as a path of multiple edges.

Consider the store-and-forward paradigm. With delay constraint $D$, any packet that traverses from $s$ to $d$ through a path longer than $D$ hops is deemed useless. Without loss of generality, we assume $D \leq |E|$. Otherwise, the problem collapses back to the classic delay-unconstrained problem since all paths have length $\leq |E|$. We also assume $\text{In}(s) = \emptyset$ and $\text{Out}(d) = \emptyset$. For any integer $k$, we define $\{1, k\} \triangleq \{1, 2, \cdots, k\}$ and define $[1, \infty)$ as the set of positive integers.

Let $P_D$ denote the collection of all $s$-to-$d$ paths of length $\leq D$ hops. Obviously $P_D$ is finite. When only store-and-forward is allowed, the largest delay-constrained routing capacity, denoted by $R^*_{\text{route}}$, can be computed by the following LP problem:

$$R^*_{\text{route}} \triangleq \max_{\{x_P \geq 0: P \in P_D\}} \sum_{P \in P_D} x_P$$

subject to $\forall e \in E$, $\sum_{P: P \ni e, P \in P_D} x_P \leq c_e$,

which consists of $|E|$ inequalities and $|P_D|$ non-negative variables $\{x_P\}$ indexed by $P \in P_D$, where each $x_P$ represents the communication rate assigned to a delay-respecting path $P$.

The objective (1) is the sum of the throughput sent over the $|P_D|$ paths, and (2) imposes that the sum rate of all paths using an edge $e$ must not exceed $c_e$. However, since $|P_D|$ grows exponentially with respect to $|G| \triangleq |V| + |E|$, the above LP characterization is not easily computable for large networks. To address the above concern of complexity, [45] proposes an equivalent alternative LP that can compute the optimal value of (1)–(2) in polynomial time of $|G|$. Since the results in [45] were stated without detailed proofs, we provide the details of [45] and the corresponding proofs in Appendix A.

By converting (1)–(2) to its dual problem, $R^*_{\text{route}}$ can also be computed by the following cut-based LP problem.

$$\min \{y_e \geq 0: e \in E\} \sum_{e \in E} y_e c_e$$

subject to $\forall P \in P_D$, $\sum_{e \in e \in P} y_e \geq 1$.

One way to interpret this dual problem is to treat the variable $y_e$ as the “cut strength” of a given edge $e$ such that the cumulative cut strength experienced by each path is no less than 1.

Note that if we replace $P_D$ by $P_\infty$, the latter of which contains all paths regardless of their lengths, then one can prove, by the standard proof techniques used in the max-flow/min-cut theorem, that for any given network instance, one of the

\[3\] It is sometimes called the hop-count-constrained max-flow problem.
minimizing \( \{y^*_e\} \) of (3)–(4) satisfies \( y^*_e \in \{0, 1\}, \forall e \in E \). As a result, we can impose the integrality constraint \( y_e \in \{0, 1\} \) to the LP problem (3)–(4) without changing the objective value. Solving (3)–(4) with the integrality constraint is equivalent to finding the minimum edge cut (those \( e \) with \( y^*_e = 1 \)). The equivalence between (1)–(2) and (3)–(4) with integrality constraints is the well-known max-flow/min-cut theorem.

However, with \( \mathcal{P}_D \) for some finite \( D \), the minimizing \( \{y^*_e\} \) can sometimes be fractional. Therefore, the delay-constrained \( \mathcal{R}^*_D \) is now characterized by some kind of soft min-cut, a unique feature of the delay-constrained setting. Examples of \( y^*_e \) being fractional can be found in the end of Section IV-B and in [24].

Also observe that there are \( |\mathcal{P}_D| \) inequalities in (4), which grows exponentially with respect to \( |G| \). In Appendix A, we have also derived a new polynomial-time version of the dual problem (3)–(4).

C. A simple comparison to RLNC

To illustrate the change of landscape when focusing on a delay-constrained setting, we provide a simple example for which the RLNC scheme [18] is no longer throughput optimal\(^4\) when there is a hard delay constraint.

Consider the network in Fig. 1(a). The min-cut value between \( s \) and \( d \) is 2, which implies the existence of (at least) one pair of edge-disjoint paths (EDPs). There are actually two possible pairs of EDPs, see Figs. 1(b) and 1(c), respectively. Assume each edge incurs a unit delay. If there is no delay constraint, we can sustain throughput 2 by routing the packets either through Fig. 1(b) or through Fig. 1(c). However, with delay constraint \( D = 3 \), only the two paths in Fig. 1(c) can be used to transmit information at rate 2. For comparison, one path in Fig. 1(b) has 4 hops, and the information transmitted along that path will expire before arriving at \( d \).

We now apply RLNC to Fig. 1(a) while assuming a sufficiently large finite field \( GF(q) \) is used, say \( GF(3) \). At each time \( t \), we send two information packets \( X_t \in GF(q) \) and \( Y_t \in GF(q) \) along the edges \( (s, v_2) \) and \( (s, v_1) \), respectively. Due to the unit-delay incurred in edge \((s, v_1)\), at time \( t \), we can send \( Y_{t-1} \) along \((v_1, v_2)\). Node \( v_2 \) can now perform RLNC. We can assume, without loss of generality, that at time \( t \) node \( v_2 \) sends \( M_{v_2d}^{(t)} = X_{t-1} + Y_{t-2} \) and \( M_{v_2v_3}^{(t)} = X_{t-1} + 2Y_{t-2} \) along \((v_2, d)\) and \((v_2, v_3)\), respectively. Following similar derivation, by the end of time \( t \), destination \( d \) should have received \( M_{v_3d}^{(t)} = X_{t-2} + 2Y_{t-3} \) and \( M_{v_3v_4}^{(t)} = X_{t-1} + Y_{t-2} \).

Since \( s \) starts to send \( Y_t \) and \( X_t \) at time \( t \) for all \( t \geq 1 \), we set \( X_t = Y_t = 0 \) for all \( t \leq 0 \). From the above derivation, by the end of time 3, \( d \) has received \( M^{(3)}_{v_3d} = X_1 + 2Y_0 = X_1 \) and \( M^{(3)}_{v_3v_4} = X_2 + Y_1 \). Recall that \( D = 3 \). Therefore, \( d \) is interested in decoding both \( X_1 \) and \( Y_1 \), which were sent by \( s \) at the “beginning” of time 1 (3 slots earlier). One can verify that knowing \( M^{(3)}_{v_3d} = X_1 \) and \( M^{(3)}_{v_3v_4} = X_2 + Y_1 \) is not sufficient for \( d \) to decode the desired \( X_1 \) and \( Y_1 \) since the value of \( Y_1 \)

\(^3\)Since RLNC is a randomized solution, a more rigorous question to ask is how large is the probability that RLNC achieves the optimal rate. We use the statement “RLNC is suboptimal” in the sense that \( \text{Prob}(\text{RLNC achieves optimal rate}) \) is quite small when a large finite field is used.

\(^4\)Here we use the convention that \( X_t = 0 \) for \( t \in [T + 1, T + D - 1] \).

III. PROBLEM FORMULATION

We follow the network model introduced in Section II-B. Namely, the network is modeled as a directed cyclic graph \( G = (V, E) \) with each edge \( e \in E \) having capacity \( c_e \) and incurring unit delay. The amount of data is measured in packets, where each packet is assumed to be chosen from \([0, q - 1]\), or equivalently each packet has \( \log_2(q) \) bits. We assume that \( q \) is a sufficiently large fixed number and in the linear network coding literature \( q \) is sometimes assumed to be a power of a prime. We can now define the delay-constrained capacity of transmitting from a single source \( s \) to a destination \( d \).

**Definition 1:** Given any network \( G = (V, E) \) and any scalar \( R > 0 \), define the message symbol set \( \mathcal{X} \) and the edge symbol set \( \mathcal{M}_e \) by

\[
\mathcal{X} = \{1, \ldots, 2^{|\log_2(q)R|}\},
\]

\[
\mathcal{M}_e = \{1, \ldots, 2^{|\log_2(q)c_e|}\}.
\]

We say the delay-constrained rate \( R \) is feasible,\(^5\) if for any given time span \( T \), we have (i) \( T \) message symbols \( X_t \in \mathcal{X}, \forall t \in [1, T] \); (ii) \( (T + D - 1) \cdot |E| \) edge-encoding functions:

For all \( t \in [1, T + D - 1] \) and \( e \in E \), if \( \text{tail}(e) = s \) then

\[
M_{e}^{(t)} = f_{e,t}(\{X_{\tau} : \tau \in [1, t]\}) \in \mathcal{M}_e,
\]

and if \( \text{tail}(e) \neq s \) then

\[
M_{e}^{(t)} = f_{e,t}(\{M_{\tau}^{(r)} : \tau \in \text{In}(\text{tail}(e)), \tau \in [1, t - 1]\}) \in \mathcal{M}_e;
\]

and (iii) \( T \) decoding functions:

\[
\hat{X}_t = f_{\text{dec},t}(\{M_{\tau}^{(r)} : e \in \text{In}(d), \tau \in [1, t + D - 1]\}) \in \mathcal{X};
\]

The network model considered herein is noiseless. For noisy networks, the traditional \((e, n)\) terminology may be needed to formally define the delay-constrained capacity.
such that \( X_t = \tilde{X}_t \) for all \( t \in [1, T] \) for all possible\(^7\) realization of \( X_t = x_t \in \mathcal{X} \).

**Definition 2:** The delay-constrained capacity \( R_{\text{NC}}^* \) is the supremum of all feasible \( R \).

It is worth emphasizing that by the above definitions \( R_{\text{NC}}^* \) is a function of the packet size \( q \). Since routing can be considered as a special example of network coding, one can easily prove the following inequality

\[
R_{\text{route}}^* \leq \lim_{q \to \infty} R_{\text{NC}}^*.
\]

(8)

where \( R_{\text{route}}^* \) is defined in (1). On the other hand, it is possible to have \( R_{\text{NC}}^* < R_{\text{route}}^* \) for some fixed small \( q \). The reason is as follows. The definition of \( R_{\text{route}}^* \) in (1)-(2) is the maximum routing capacity in a splittable flow setting since the values \( \{x_P : P \in \mathcal{P}_D \} \) are allowed to be fractional. On the other hand, for finite \( q \), the definition of \( R_{\text{NC}}^* \) can be viewed as the maximum NC capacity in an unsplittable setting where the size of \( q \) controls the granularity of information splitting. Since splitting the flows can increase the achievable throughput, we may have \( R_{\text{NC}}^* < R_{\text{route}}^* \) when \( q \) is small. On the other hand, when \( q \to \infty \), the \( R_{\text{NC}}^* \) now corresponds to a splittable setting and we thus have (8).

IV. MAIN RESULTS

In this section, we state our main results: (i) An equivalent time-expanded multiple unicast formulation; (ii) The NC gain over routing; and (iii) A new delay-constrained capacity upper bound.

A. An equivalent multiple-unicast formulation

For any finite network \( G = (V, E) \), the time-expanded graph over a time horizon \( t = [1, \tau] \) can be defined as follows.

**Definition 3 (30, Section IV):** For any fixed integer \( \tau \geq 1 \) and any fixed network \( G = (V, E) \), the time-expanded graph \( \tilde{G}^\tau = (\tilde{V}^\tau, \tilde{E}^\tau) \) contains \( \tau \cdot |V| \) nodes, each node being labeled by a pair \( [v, t] \) for all \( v \in V \) and \( t \in [1, \tau] \). The edge set \( \tilde{E}^\tau \) can be described/constructed as follows.

1) For each \( e = (u, v) \in E \) with the corresponding capacity being \( c_e \) and for each \( t \in [1, \tau - 1] \), there exists an edge \( \tilde{e}^{[\tau]} \in \tilde{E}^\tau \) that connects \([u, t] \) and \([v, t + 1] \). The capacity of the edge \( \tilde{e}^{[\tau]} \) is set to \( c_e \).

2) For each node \( u \in V \) and \( t \in [1, \tau - 1] \), there exists an edge \( \tilde{e}^{[\tau]} \in \tilde{E}^\tau \) that connects \([u, t] \) and \([u, t + 1] \). The capacity of the edge \( \tilde{e}^{[\tau]} \) is set to

\[
t \cdot \left( \sum_{e \in \text{In}(u)} c_e \right).
\]

if \( u \neq s \); and is set to \( t \cdot \log_q \left( \frac{|g|}{|r|} \right) \) if \( u = s \). The capacity of \( \tilde{e}^{[\tau]} \) is to ensure that node \([u, t] \) can pass all the “external/incoming” information it has accumulated in the past (i.e., during time \([1, t]\)), directly to \([u, t + 1] \).

With the above definition, we present an equivalent formulation of the delay-constrained capacity.

**Proposition 1:** A delay-constrained rate \( R \) is feasible if and only if for any \( T \), the time-expanded graph \( \tilde{G}^{T+D} \) can sustain simultaneously \( T \) multiple unicast flows, where each unicast flow is indexed by \( t \in [1, T] \) and is from node \([s, t] \) to node \([d, t + D] \) and has zero-error rate \( R \).

**Proof:** The only if direction “\( \Rightarrow \)” : If delay-constrained rate \( R \) is feasible in \( G \), then for any given \( T \) we can design a feasible multiple-unicast scheme in \( \tilde{G}^{T+D} \) in the following way. For any \( t \) value, we let the \([s, t]\)-to-\([s, t + 1]\) edge carry all the symbols \( \{X_t : \tau \in [1, t]\} \), and let the \([u, t]\)-to-\([u, t + 1]\) edge, \( u \neq s \), carry all the symbols \( \{M_{\tilde{r}} : \tilde{r} \in \text{In}(u), \tau \in [1, t + 1]\} \). In this way, the \([u, t]\)-to-\([u, t + 1]\) edge can compute and transmit \( M^{(t)}_{(u, v)} \) described in (5) and (6), and the destination \([d, t + D] \) can compute \( \tilde{X} \) described in (7). The multiple-unicast zero-error rate \( R \) is thus feasible in \( \tilde{G}^{T+D} \).

The if direction “\( \Leftarrow \)” : Suppose for any given \( T \) we can design a multiple-unicast scheme in \( \tilde{G}^{T+D} \) with zero-error rate \( R \). For all \((s, u) \in E \), we then choose the \( M^{(t)}_{(s, u)} \) message in \( G \) to be the message in \( \tilde{G}^{T+D} \) that is sent from \([s, t]\)-to-\([u, t + 1]\). Since for any given \( t \), the values of \( \{X_t : \tau \in [1, t]\} \) determine uniquely the message along the \([s, t]\)-to-\([u, t + 1]\), the chosen \( M^{(t)}_{(s, u)} \) must be of the form of (5).

For any \( t \) and any \((v, u_2) \in E \) satisfying \( v \neq s \), we choose the \( M^{(t)}_{(v, u_2)} \) message in \( G \) to be the message in \( \tilde{G}^{T+D} \) that is sent from \([v, t]\)-to-\([u_2, t + 1]\). Since the edge set

\[
\{(u_1, \tau) : \forall \tau \in [1, t + 1], (u_1, v) \in E\}
\]

is a cut in \( \tilde{G}^{T+D} \) separating \( \{X_t : t \in [1, T]\} \), the chosen \( M^{(t)}_{(v, u_2)} \) must be of the form of (6).
For any \( t \), we choose the decoder function \( f_{\text{DEC}, t} \) in \( G \) to be the decoder function in \( G^{[T+D]} \) used by destination \([d, t + D]\).
Since the edge set
\[
\{(u, \tau), [d, \tau + 1) : \forall \tau \in [1, t + D - 1], (u, d) \in E\}
\]
is a cut in \( G^{[T+D]} \) separating \([X_t : t \in [1, T]]\) from destination \([d, t + D]\), the chosen decoder \( f_{\text{DEC}, t} \) must be of the form of (7). Since the chosen encoders/decoders can support \( T \) coexisting unicast traffic in \( G^{[T+D]} \) with rate \( R \), the network coding solution in \( G \) can support delay-constrained capacity. The proof is thus complete.

The above connection between the delay-constrained capacity with the multiple-unicast capacity in a time-expanded network will later be used to derive a new upper bound on delay-constrained capacity. Also, being equivalent to the multiple-unicast capacity in a time-expanded network, it is a cut in \( G^{[T+D]} \). Hence, in the end of time 6, node \( Y_1 \) with delays \( D = 6 \) can decode both \( X_2 \) and \( Y_0 \) and has already received \( M^{(7)}_{v_{8d}} = X_2 + Y_1 \) and \( M^{(7)}_{v_{1d}} = 2X_2 + Y_1 \). Then \( d \) can use \( M^{(6)}_{v_{1d}} = 2X_2 + Y_1 \) and \( M^{(6)}_{v_{8d}} = X_2 + Y_1 \) to decode both \( X_2 \) and \( Y_1 \). However, \( Y_2 \) still cannot be decoded since \( M^{(7)}_{v_{1d}} = 2X_3 + Y_2 \) is corrupted by \( X_3 \), which has not been decoded yet.

On the other hand, we can perform the following optimal NC instead of the simple addition in (10). That is, instead of “adding” the two incoming packets, we now “subtract” \( M^{(t-1)}_{v_{10v4}} \) from \( M^{(t)}_{v_{3v4}} \):

\[
\text{Optimal: } M^{(t)}_{v_{4d}} = M^{(t-1)}_{v_{3v4}} - M^{(t-1)}_{v_{10v4}} = Y_{t-5}.
\]

Destination \( d \) can now decode \( X_1 \) and \( Y_1 \) from \( M^{(6)}_{v_{8d}} = X_1 + Y_0 = X_1 \) and \( M^{(6)}_{v_{1d}} = Y_1 \) within the delay constraint.

An astute reader may notice that in the end of time 7, \( d \) has received \( M^{(7)}_{v_{1d}} = X_2 + Y_1 \) and \( M^{(7)}_{v_{8d}} = Y_2 \), where \( M^{(7)}_{v_{8d}} \) is “corrupted” by \( Y_1 \). Nonetheless, \( d \) can remove \( Y_1 \) in the end of time 7 since \( d \) has decoded \( Y_1 \) in the end of time 6. The above argument can be used to prove that \( d \) can decode \( X_1 \) and \( Y_1 \) (injected in the beginning of time \( t \)) by the end of time \( t+5, \forall t \geq 1 \). The \( D = 6 \) constraint is met. Since \( \text{min-cut}(s, d) = 2 \) in Fig. 2, we have the delay-constrained NC capacity being \( R^*_{\text{NC}} = 2 \) packets per slot.

We then apply (3)–(4) to Fig. 2 and derive \( R^*_{\text{route}} = 1.5 \). The corresponding minimizing \( y^*_{e} \) are: \( y^*_{v_1} = y^*_{v_2v_3} = y^*_{v_6v_4} = 0.5 \) and all other \( y^*_{e} = 0 \). The optimal routing solution in (1)–(2) will assign rate 0.5 to three different paths: \( su_{v_5v_6}vu_{v_2v_3}v_4d, su_{v_1v_2}v_3vu_8v_6d, \) and \( su_{v_5v_6}vu_{v_2v_3}v_4d, su_{v_1v_2}v_3vu_8v_6d, \) and \( su_{v_5v_6}vu_{v_2v_3}v_4d, su_{v_1v_2}v_3vu_8v_6d, \), and this example shows that NC strictly outperforms optimal routing even for the single-unicast setting.

### C. How large can the NC gain be?

In the previous example, the NC throughput gain over routing is \( \frac{R^*_{\text{NC}}}{R^*_{\text{route}}} = \frac{2}{1.5} \). An interesting open question is what is the largest NC gain in a single-unicast delay-constrained setting? Specifically, we are interested in quantifying

\[
\sup_{G \in \mathcal{G}, \forall D \in [1, \infty)} \text{gain}_v(G, D) \quad (12)
\]
where $G_{s-u}$ contains all possible network instances\textsuperscript{10} with single-unicast (s-u) traffic, and $\text{gain}_{s-u}(G, D) \triangleq \frac{R_{\text{NC}}}{R_{\text{route}}}$ is the single-unicast NC gain over routing in $G$ with delay constraint $D$.

For comparison, one can easily prove that the delay-constrained NC gain can be unbounded for the single-multicast (s-m) networks and for the multiple-unicast (m-u) networks, denoted by $G_{s-m}$ and $G_{m-u}$, respectively. Namely,\textsuperscript{11}

$$\sup_{G \in G_{s-m} \cap D} \text{gain}_{s-m}(G, D) \geq \sup_{G \in G_{s-m}} \text{gain}_{s-m}(G, 2) = \infty$$

(13)

and

$$\sup_{G \in G_{m-u} \cap D} \text{gain}_{m-u}(G, D) \geq \sup_{G \in G_{m-u}} \text{gain}_{m-u}(G, 3) = \infty$$

(14)

where the equality in (13) follows from the combination network construction in [34] and the equality in (14) follows from the extended butterfly construction in [17]. For the multiple unicast scenarios, we define $\text{gain}_{m-u}(G, D) \triangleq \frac{R_{\text{NC}}}{R_{\text{route}}}$ where we slightly abuse the notation and use $R_{\text{NC}}$ and $R_{\text{route}}$ to describe the perfectly fair (or equivalently equal-rate) capacity of all the coexisting flows, also see the feasible throughput definition in [15] and the concept of symmetric capacity in [23].

Nonetheless, the proofs of (13) and (14) cannot be applied to the single-unicast setting since they rely heavily on the fact that there are multiple destinations so that different destinations can either capitalize the diversity gain (for single multicast) or smartly cancel the interference of the other coexisting flows (for multiple unicast). These types of gains do not exist when there is only one destination\textsuperscript{11} in the network.

Our best understanding of (12) is summarized as follows.

**Proposition 2:** For any $0 < \epsilon < 1$, there exists a network $G \in G_{s-u}$ and delay constraint $D$ satisfying

$$\frac{R_{\text{NC}}}{R_{\text{route}}} > 2 - \epsilon.$$ \hspace{1cm} (15)

Proposition 2 shows that for delay-constrained unicast traffic, NC gain over routing can be arbitrarily close to 2. A proof of Proposition 2 is provided in Appendix B.

In a broad sense, $R_{\text{route}}$ characterizes the maximum number of EDPs with length $\leq D$ hops, along which we can “squeeze through” $R_{\text{route}}$ packets before expiration. Therefore, at least heuristically, any additional packets sent over the network (other than the original $R_{\text{route}}$ packets) are either dependent or experiencing too long delay. Proposition 2 implies a rather counter-intuitive result:

*With carefully-designed NC, those additional “useless” packets (either dependent or experiencing too long delay) can help us double the number of independent packets that can be decoded by $d$ within the delay constraint.*

In terms of quantifying the largest possible NC gain in (12), Proposition 2 proves that

$$2 \leq \sup_{G \in G_{s-u}, D \in [1, \infty)} \text{gain}_{s-u}(G, D).$$

(16)

In a recent work [19, Theorem 1], it has been shown that the NC gain over routing satisfies that for any fixed $D$

$$\sup_{G \in G_{s-u}} \text{gain}_{s-u}(G, D) \leq 8 \log_e(D + 1).$$

(17)

The gap between the lower bound (15) and the upper bound (16) is still substantial. Two observations can be made regarding the above upper and lower bound pair. Firstly, (16) goes to infinity when $D \to \infty$. Therefore (16) may not be suitable to directly upper bound (12).

On the other hand, (16) suggests implicitly that when searching for network instances of large $\text{gain}_{s-u}(G, D)$, it is critical to consider large delay requirement $D$ and even larger network diameter\textsuperscript{12} $\text{dia}(G)$. This suggestion is consistent with our findings, for which our construction of $(G, D)$ with $\text{gain}_{s-u}(G, D) \to 2$ indeed has large $D$ and large $\text{dia}(G)$. Since the traditional ways of constructing network instances with large single-multicast and large multiple-unicast gains [17], [34] all have very short $\text{dia}(G)$, it is less likely that those construction methods can lead to network instances with large $\text{gain}_{s-u}(G, D)$.

D. Upper bounding the NC capacity

The delay-constrained $R_{\text{route}}^*$ naturally serves as a lower bound on $R_{\text{NC}}^*$, assuming the underlying alphabet size $q$ is sufficiently large. See (8). We now present a new upper bound on $R_{\text{NC}}^*$.

**Proposition 3:** The following integer programming (IP) problem computes an upper bound $\text{UB}_{\text{NC}}$ on $R_{\text{NC}}^*$ for any alphabet size $q$:

$$\begin{align*}
\min_{\{y_e, e \in E\}} & \sum_{e \in E} y_e c_e \\
\text{subject to} & \forall P \in \mathcal{P}_D, \sum_{e \in P} y_e \geq 1.
\end{align*}$$

(18)

$$\forall e \in E, y_e \in \{0, 1\}.$$ \hspace{1cm} (19)

**Proof:** Consider any delay-constrained rate $R$ that is feasible. By Proposition 1, for any $T$ value, the time expanded graph $\mathcal{G}_{[T+D]}$ can sustain $T$ simultaneous unicast flows from $[s, t]$ to $[d, t + D]$ for all $t \in [1, T]$ with individual rate $R$.

For any given IP solution $\{y_e\}$ satisfying (18)-(19), we construct an edge set in the time expanded graph $\mathcal{G}_{[T+D]}$ by

$$\mathcal{E}_{\text{cut}} \triangleq \{(u, v) \in \mathcal{E} : \forall (u, v) \text{ s.t. } y_{(u, v)} = 1, \forall \tau \in [1, T + D - 1]\}.$$ \hspace{1cm} (20)

\textsuperscript{10}More precisely, a network instance should contain a 5-tuple $(G, \{c_e\}, s, d, q)$ where $G$ is the network topology, $\{c_e\}$ is the edge capacity, $s$ and $d$ are the source/destination nodes, and $q$ is the alphabet size under consideration. For notational simplicity, we use “$G \in G_{s-u}$” as shorthand for the more precise expression “$(G, \{c_e\}, s, d, q) \in G_{s-u}$.”

\textsuperscript{11}In the single-unicast setting, one needs to consider a different type of interference. That is, optimal NC needs to remove the corruption caused by future, not-yet decoded packets within the same flow. See the detailed discussion of the suboptimal RLNC choice (10) versus the optimal choice (11). Such a new notion of interference is strongly coupled with the time-axis and calls for the development of new analysis tools.

\textsuperscript{12}One can prove that for any fixed $G$, if $D \geq \text{dia}(G)$, then we always have $\text{gain}_{s-u}(G, D) = 1$. The reason is that $\text{dia}(G)$ is an upper bound of all paths from $s$ to $d$. Therefore if $D \geq \text{dia}(G)$ then any path will meet the delay constraint, which is as if $D = \infty$, and the single-unicast network coding gain for $D = \infty$ is known to be 1. This observation shows that a combination of large $D$ but small $\text{dia}(G)$ will not yield large $\text{gain}_{s-u}(G, D)$. 


By interpreting (18) in the time expanded graph $G^{[T+D]}$, one can verify that the above choice of $E_{\text{cut}}$ is an edge cut separating $[s,t]$ from $\{(d,t+D) : \forall t \in [1,T]\}$ for all $t \in [1,T]$. By the generalized network-sharing bound in [21], we thus have

$$T \cdot R \leq \sum_{\tau \in E_{\text{cut}}} c_{\tau}$$

(21)

$$= (T + D - 1) \sum_{e} y_e c_e$$

(22)

where $\tau$ represents an edge in the time expanded graph $G^{[T+D]}$ and $c_{\tau}$ is the corresponding edge capacity; (21) follows from that the sum of the rate $T$ coexisting flows is no larger than the generalized cut set value [21]; and (22) follows from the definition of $E_{\text{cut}}$ in (20).

Ineq. (22) implies $R \leq \frac{T + D - 1}{\sum_{e} y_e c_e}$ for all $T$. By letting $T \to \infty$ and by finding an IP solution $\{y_e\}$ that minimizes (17), the proof is complete.\]

Comparing Proposition 3 with (3)–(4), we see that adding the integer condition (19) to the minimization problem turns $R_{\text{route}}$, a lower bound on $R_{\text{NC}}$, to an upper bound $UB_{\text{NC}}$ on $R_{\text{NC}}$. Proposition 3 also implies that for any network instance, if the minimizing $y^*_e$ of (3)–(4) is integral, then the lower and upper bounds match and we have fully characterized the delay-constrained unicast NC capacity: $R_{\text{route}} = R_{\text{NC}} = UB_{\text{NC}}$. On the other hand, for any network instance in which $R_{\text{NC}} > R_{\text{route}}$, e.g., Fig. 2, the $y^*_e$ of (3)–(4) must be fractional, which is consistent with our observation in the end of Section IV-B.

To illustrate Proposition 3, we apply the upper bound to three network instances: Fig. 3(a), Fig. 2, and Fig. 3(b), respectively. We first focus on the IP problem generated by applying Proposition 3 to Fig. 3(a). Simple counting arguments can be used to show that the following assignment of $y^*_e$ is an optimal solution of the corresponding IP problem: $y^*_e = 1$ if $e \in \{(s,v_2),(v_2,d)\}$ and $y^*_e = 0$ for all other $e$. As a result, we have $R_{\text{NC}} \leq 1 + 1 = 2$ for Fig. 3(a). The upper bound in Proposition 3 thus shows that the high-capacity edges $c(s,v_1) = c(v_1,v_2) = c(v_2,v_3) = c(v_3,d) = 3$ do not increase the delay-constrained capacity regardless how the network code is designed. Also note that the edge set $\{(s,v_2),(v_2,d)\}$ does not separate $s$ from $d$ since $s$ can still reach $d$ through $sv_1v_2v_3d$. On the other hand, the edge set $\{(s,v_2),(v_2,d)\}$ “cuts” all paths of length $\leq D = 3$.

We now apply Proposition 3 to Fig. 2 and generate the corresponding IP problem. Some simple observation can be used to show that for this particular IP problem, the minimizing $\{y^*_e\}$ is not unique. One minimizing solution is $y^*_e = 1$ if $e \in \{(v_4,d),(v_8,d)\}$ and $y^*_e = 0$ for all other $e$. As a result, we have $R_{\text{NC}} \leq 1 + 1 = 2$ for Fig. 2. Proposition 3 is tight for Fig. 2 since Section IV-B proves that NC can indeed achieve the delay-constrained capacity 2 packets per time slot.

Unfortunately, Proposition 3 can be loose\(^{13}\) in some scenarios. For example, consider the network in Fig. 3(b). Comparing Figs. 2 and 3(b), the only difference is that the path $v_1v_3v_1v_4$ in Fig. 2 is now lengthened to $v_1v_3v_1v_4$. We apply Proposition 3 to Fig. 3(b) and generate the corresponding IP problem. One minimizing solution of the IP problem is $y^*_e = 1$ if $e \in \{(v_4,d),(v_8,d)\}$ and $y^*_e = 0$ for all other $e$. As a result, we have $R_{\text{NC}} \leq 1 + 1 = 2$ for Fig. 3(b). However, as will be shown later, the network in Fig. 3(b) has $R_{\text{NC}} = R_{\text{route}} = 1.5$. Namely, regardless how the network code is designed, the best achievable $R_{\text{NC}}$ is no larger than $R_{\text{route}}$. The upper bound in Proposition 3 fails to provide a tight bound and is loose for the network in Fig. 3(b).

In the following, we provide an improved upper bound by directly describing the upper bound in the time-expanded network.

**Proposition 4:** For any fixed integer $L > 0$, we define a binary mapping $h : E \times [0,L-1] \to \{0,1\}$ where $E$ is the collection of all edges in $G$. Once the mapping $h$ is fixed, we define the following edge set

$$E_h \triangleq \{(u,t),(v,t+1) : \forall e = (u,v) \in E, \forall t \in [1,L+D-1] \text{ s.t. } h(e, \text{ mod } (t,L)) = 1\}$$

(23)

in the time expanded graph $G^{[L+D]}$.

If for any $t \in [1,L]$ the edge set $E_h$ is always an edge cut separating $[s,t]$ from $[d,t+D]$, then we have a new upper bound

$$R_{\text{NC}} \leq \sum_{e} \left(\frac{\sum_{i=0}^{L-1} h(e,i)}{L}\right) c_e$$

(24)

**Remark:** An upper bound similar to Proposition 4 is discovered independently in [19, Theorem 3]. The following corollary shows that Proposition 4 is a strict generalization of Proposition 3.

**Corollary 1:** The upper bound in Proposition 4 with $L = 1$ is equivalent to the upper bound in Proposition 3.

The proofs of Proposition 4 and Corollary 1 are relegated to Appendix G.

We now use Proposition 4 to prove that $R_{\text{NC}} = R_{\text{route}} = 1.5$ for the network in Fig. 3(b). This shows that Proposition 4 can be strictly tighter than Proposition 3. It also shows that when the path $v_1v_3v_1v_4$ in Fig. 2 is getting longer, the side information carried through that path is getting older and thus

\(^{13}\) An example in [19] shows that the gap between the upper bound in Proposition 3 and the lower bound $R_{\text{route}}$ can approach infinity.
can no longer be used to remove the interference along the \((v_4, d)\) edge. The \(R_{NC}\) capacity will thus reduce from 2 to 1.5 and the NC gain in Fig. 2 disappears.

To apply Proposition 4 to Fig. 3(b), we choose \(L = 2\) and set

\[
h(e, 0) = \begin{cases} 1 & \text{if } e \in \{(s, v_1), (v_4, d)\} \\ 0 & \text{for all other } e \end{cases} \quad (25)
\]

\[
h(e, 1) = \begin{cases} 1 & \text{if } e = (v_2, v_3) \\ 0 & \text{for all other } e \end{cases} \quad (26)
\]

In Appendix H, we have proven that for the network in Fig. 3(b) with the binary mapping \(h(e, t)\) given in (25)–(26), the edge set \(E_h\) is always an edge cut in the time expanded graph \(G_{L+D}[E]\) that separates \([s, t]\) from \([d, t + D]\) for all \(t \in [1, L]\). As a result, by Proposition 4 we have

\[
R_{NC}^e \leq \sum_{e \in \{(s, v_1), (v_4, d), (v_2, v_3)\}} \frac{1}{2} \cdot c_e = 1.5.
\]

V. VARIOUS OTHER RESULTS ON DELAY-CONSTRAINED CAPACITY

This section contains various other results on delay-constrained capacity, including the unboundedness of the penalty of using RLNC and an example showing that Edmonds’ tree packing theorem no longer holds for the delay-constrained setting. In the end of this section, we also discuss how our results can be applied to cyclic networks verbatim.

A. The penalty of using RLNC can be unbounded

In Section II-C, we have shown that even when considering only a single unicast flow, the best RLNC scheme may have strictly worse performance than \(R_{route}^*\) when the traffic is delay-constrained. Similar to our investigation in Section IV-C where we studied the largest possible NC gain \(R_{NC}^e/R_{RLNC}^e\) in this section we study the largest possible penalty of using RLNC when compared to the routing solution. That is, we are interested in quantifying the following RLNC penalty:

\[
\text{Largest RLNC Penalty} \triangleq \sup_{G \in \mathcal{G}, \alpha \in [1, \infty)} \frac{R_{route}^*}{R_{RLNC}^*}. \quad (27)
\]

By the example in Section 5, we know that the largest RLNC penalty is no less than 2.

To quantify the RLNC penalty in (27), we first formally define the class of NC being considered. Specifically, we focus exclusively on the unit-edge-capacity network, i.e., \(c_e = 1\) for all \(e \in E\). All the vectors in our discussion, unless specified otherwise, are column vectors.

For any given integer rate \(R\), the RLNC scheme is defined as follows. There are \(R\) message symbols in each time slot, and they are defined as a vector \(\bar{X}(t) = (X_1(t), X_2(t), \ldots, X_R(t))^T \in (GF(q))^R\) where \(GF(q)\) is the finite field of order \(q\) and we assume \(q\) is a power of a prime. The source \(s\) is associated with \(|\text{Out}(s)|\) precoding vectors \(\tilde{\beta}_{s, e}, \forall e \in \text{Out}(s)\) and each is of dimension \(R\). Each intermediate node \(v \in V\setminus\{s, d\}\) is associated with \(|\text{In}(v)| \cdot |\text{Out}(v)|\) local coding coefficients \(\beta_{e_1, e_2} \in GF(q), \forall e_1 \in \text{In}(v), e_2 \in \text{Out}(v)\). At every time slot \(t\), source \(s\) sends the coded symbols

\[
M_{e_1}^{(t)} = \beta_{e_1, e_2} \bar{X}(t) \quad (28)
\]

along its outgoing edges \(e \in \text{Out}(s)\); and any intermediate node \(v \in V \setminus \{s, d\}\) sends the coded symbols

\[
M_{e_2}^{(t)} = \sum_{e_1 \in \text{In}(v)} \beta_{e_1, e_2} M_{e_1}^{(t-1)} \quad (29)
\]

along all its outgoing edges \(e \in \text{Out}(v)\).

During the design phase, the RLNC scheme chooses the precoding vectors \(\tilde{\beta}_{s, e}, \forall e \in \text{Out}(s)\) and the local coding coefficients \(\beta_{e_1, e_2}, \forall e_1 \in \text{In}(v), e_2 \in \text{Out}(v), v \in V \setminus \{s, d\}\) independently and uniformly randomly from \((GF(q))^R\) and from \(GF(q)\), respectively. We assume that after the design phase, both \(\tilde{\beta}_{s, e}\) and \(\beta_{e_1, e_2}\) are made known to all network nodes, including \(s\) and \(d\). With that knowledge, the network can perform RLNC based on (28) and (29). We assume that the choices of \(\tilde{\beta}_{s, e}\) and \(\beta_{e_1, e_2}\) are fixed after the initial design phase and they do not change over time \(t\).

**Definition 4:** We say RLNC can support an integer delay-constrained rate \(R\) if the probability that at a random design phase, destination \(d\) can decode \(\bar{X}(t)\) by the end of time slot \(t + D - 1, \forall t \in [1, \infty)\), approaches 1 when the underlying finite field size \(q\) is sufficiently large. The largest supportable rate of RLNC is denoted by \(R_{RLNC}^*\).

We now have the following result.

**Proposition 5:** The penalty of using RLNC can be arbitrarily large. That is,

\[
\sup_{G \in \mathcal{G}, \alpha \in [1, \infty)} \frac{R_{route}^*}{R_{RLNC}^*} = \infty.
\]

The detailed proof of Proposition 5 is relegated to Appendix I, which will be based on explicit network construction. The very first idea of constructing a network with large penalty (27) is to generalize Fig. 1(a) in a way similar to Fig. 4(a). Specifically, there are 3 paths in Fig. 4(b) with length exactly equal to the delay requirement \(D = 4\). Therefore, the optimal \(R_{route}^* = 3\). On the other hand, since all 3 paths share a common node \(v_2\), in a similar way as discussed in Section II-C, the packets along the \(sv_4v_5vd\) and the \(sv_1v_2vd\) paths will be contaminated by some sort of future packets due to the random packet mixing at node \(v_2\). Note that the packets along the \(sv_2v_1v_2d\) are contaminated by the past packets, which can be automatically canceled by the packets decoded in the previous time slots.

![Fig. 4. A straightforward generalization of Fig. 1(a) with \(D = 4\). This network has \(R_{NC}^e = R_{route}^e = 3\) and \(R_{RLNC}^e = 2\).](image-url)
As a result, one might expect that Fig. 4(a) has $R_{RLNC}^* = 1$ since only the packets entering $d$ through the $(v_7,d)$ edge will be free from the corruption of the future packets. In Appendix I, we prove that Fig. 4(a) actually has $R_{RLNC}^* = 2$. As a result the RLNC penalty of Fig. 4(a) is $\frac{3}{2}$, which is even smaller than the simple example in Fig. 1(a) where the RLNC penalty is 2. The reason $R_{RLNC}^* = 2$ in Fig. 4(a) is because destination $d$ can carefully remove some of the “future interference” by intelligently combining the packets received from the three edges $(v_2,d)$, $(v_3,d)$, and $(v_7,d)$. The main challenge of proving Proposition 5 is to construct a network and prove that even with an optimal decoder at $d$ can remove the “future interferences” by intelligently combining different received packets, the resulting $R_{RLNC}$ is still arbitrarily away from the routing rate $R_{route}^*$. A simple generalization of Figs. 1(a) like the one described in 4(a) will not work.

In Appendix I, we show that for any $m \geq 2$, one can design a network for which $R_{route}^* = m$ and $R_{RLNC}^* = 1$ even with the use of an optimal decoder at destination $d$.

B. The 1-to-all scenario

In this section, we consider the 1-to-all broadcast scenario. Namely, source $s$ would like to send the same information to all destinations $V \setminus s$ at rate $R$. The tree-packing theorem in [6], [43] proves that the largest possible information-theoretic capacity $R^*$ can be achieved by pure routing without the need of NC. Namely, $R_{NC}^* = R_{route}^*$ for the 1-to-all scenario.

In this section, we will prove by an example that such an elegant result does not hold when the information is delay-constrained.

Proposition 6: Denote all the 1-to-all network scenarios by $G_{1-all}$. We have

$$\sup_{G \in G_{1-all}} \frac{R_{NC}^*}{R_{route}^*} = \infty.$$  

Proof: We first describe an example network with $\frac{R_{NC}^*}{R_{route}^*} = \frac{2}{17}$. Since our construction is based on simple modification of the $\binom{m}{2}$ combination network construction in [34], by the same analysis on the most general $\binom{m}{n}$ combination networks in [34] one can complete the proof.

Consider the network in Fig. 5(a), for which the edge capacity $c_e = 1, \forall e \in E$ and there are two parallel edges connecting $s$ and $v_7$. One can check that the min-cut values satisfy $\text{min-cut}(s,v_1) = 2$ for all $v_1 \in V \setminus s$. Edmonds’ tree packing results show that such a network can support 2 edge-disjoint spanning trees, see Fig. 5(b). Therefore when there is no delay constraint ($D = \infty$) we have $R_{NC}^* = R_{route}^* = 2$.

We now impose a delay requirement $D = 2$. It is clear that by sending $X(t), Y(t)$, and $X(t)+Y(t)$ at time $t$ along $(s,v_1), (s,v_2)$, and $(s,v_3)$, respectively, all six nodes $v_1$ to $v_6$ can decode both $X(t)$ and $Y(t)$ by the end of time $t+1$, provided $v_1$ to $v_3$ simply forwarding what they have received in the previous time slot to all its downstream neighbors. Source $s$ can send both $X(t)$ and $Y(t)$ directly to $v_7$. Since all 7 nodes $v_1$ to $v_7$ can be satisfied simultaneously, we have $R_{NC}^* = 2$.

To prove that $R_{route}^* = 1.5$, we consider how to use routing to satisfy nodes $v_4$ to $v_6$ and temporarily ignore the needs of $v_1$ to $v_3$ and $v_7$. We observe that when satisfying nodes $v_4$ to $v_6$, the routing paths must not use any of the five edges $(s,v_7), X_v, (v_1,v_2), (v_1,v_3)$. Otherwise, the paths will have length $\geq 3$, which exceeds the delay requirement $D = 2$. However, if we remove edges $(s,v_7), X_v, (v_1,v_2), (v_1,v_3)$, then the resulting network is a $\binom{5}{2}$ combination network defined in [34]. The results in [34] show that the largest possible routing rate that satisfies $v_4$ to $v_6$ simultaneously is 1.5. By revising the scheme in [34] slightly, we can also satisfy nodes $v_1$ to $v_3$ and $v_7$ with rate 1.5. As a result, Fig. 5(a) has $R_{route}^* = 1.5$.

To prove that $\frac{R_{NC}^*}{R_{route}^*}$ can be unbounded, we start from any arbitrarily given $\binom{m}{n}$ combination networks described in [34] and modify it by introducing new edges among the $n$ layer-1 nodes $(v_1,v_i)_{x \times m-1}$ for all $i \in [2,n]$. The notation $(v_1,v_i)_{x \times m-1}$ denotes $(m-1)$ parallel edges connecting $v_1$ and $v_i$. See the $\{(v_1,v_2),(v_1,v_3)\}$ edges in Fig. 5(a) for illustration. We then introduce an auxiliary node $v_{2n+1}$ that add two edges $(s,v_{2n+1}), (v_{2n+1},v_1)$ for $x \times m-1$. See the $\{(s,v_7), (v_1,v_1)\}$ edges in Fig. 5(a) for illustration. Using the same argument, a routing solution cannot use those new edges due to the delay constraint. Therefore, the 1-to-all network coding gain for the modified network is the same as the network coding gain for the original $\binom{m}{n}$ combination network. 

C. Remarks on Cyclic networks

All our results can be applied to cyclic networks verbatim. More specifically, Proposition 1 proves the equivalence between a delay-constrained unicast problem and a multiple-unicast problem in the corresponding time-expanded graph. Since the causality in time automatically converts any network (regardless being cyclic or not) into an acyclic time-expanded graph, Proposition 1 holds naturally for cyclic networks as well. Propositions 3 and 4 are proven based on the equivalent time-expanded graph (Proposition 1). As a result, they hold for cyclic networks as well. Propositions 2, 5, and 6 are proven by explicit construction of special acyclic network instances. Since acyclic networks are a special case of cyclic ones, Propositions 2, 5, and 6 hold for cyclic networks as well.
VI. CONCLUSION

This work studies the following problem: Given a hard delay constraint, how much perishable information one can send from $s$ to $d$. We have provided a new formulation that converts the delay-constrained unicast capacity problem into a delay-unconstrained multiple unicast problem. We have then proven that NC can strictly outperform optimal routing even for the single-unicast setting and the gain can be arbitrarily close to 2. Based on a time-expanded graph approach, two new delay-constrained capacity upper bounds are provided, which show that the difference between the NC capacity $R_{NC}^*$ and the routing capacity $R_{route}^*$ can be associated to the integrality gap of the LP-based minimum cut problem. We have also proven that the penalty of using RLNC versus pure routing can be unbounded and the classic Edmonds’ tree packing results no longer hold for the delay-constrained traffic. Overall, our results suggest that delay-constrained communication is fundamentally different from the well-understood delay-unconstrained one and call for investigation participation.

There are several interesting directions for further investigation. In the current setting, information bits incur a fixed amount of delay when traversing a link. It is interesting to extend the study to consider a general setting where the delay of traversing a link is also a function of the traffic volume, which better models practical scenarios involving fast-timescale system/flow dynamics.

APPENDIX A

A LOW-COMPLEXITY COMPUTATION OF THE DELAY-CONstrained ROUTING CAPACITY

The existing polynomial-complexity computation of the delay-constrained routing capacity in [45] is restated as follows.

Proposition 7 (Section IV A [45]): We can compute $R_{route}^*$ in (1)–(2) by the following flow-based LP problem with $|E| \cdot D$ non-negative variables $x_{e}^{(h)}$ for all $e \in E$ and $h \in [1, D]$:

$$\max \sum_{e \in \text{In}(d)} \sum_{h=1}^{D} x_{e}^{(h)} \quad \text{subject to} \quad \forall v \in V \setminus \{s, d\}, \forall h \in [1, D],$$

$$x_{e}^{(h)} = \sum_{e \in \text{Out}(v)} x_{e}^{(h-1)}$$

$$\forall e \in E, \sum_{h=1}^{D} x_{e}^{(h)} \leq c_e. \quad \text{(32)}$$

Here each variable $x_{e}^{(h)}$ represents the sum of all rates assigned to paths $\{P \in P_D : \text{the } h\text{-th hop of } P \text{ is } e\}$. The objective in (30) is the aggregate rate of flows that arrive at $d$ within D hops. The constraints in (31) say that the aggregate incoming flows to node $v$ with hop count $h - 1$ must be equal to the aggregate outgoing flows from node $v$ with hop count $h$; these are essentially the flow balance equations with flow travelled-distance (in hops) taken into account. The constraints in (32) are link capacity constraints. Note that in (31) we use the convention that $x_{e}^{(0)} = 0$ for all $e \in E$.

Since the proof of Proposition 7 was omitted in [45], we sketch the proof in the following for completeness.

Proof: We first prove that any solution in the LP problem (1)–(2) leads to a valid solution for the LP problem (30)–(32). This is done by setting

$$x_{e}^{(h)} = \sum_{P \in P_D : \text{the } h\text{-th hop of } P \text{ is } e} x_{P}. \quad \text{(33)}$$

With the above construction of $x_{e}^{(h)}$, one can see that (31) holds naturally and (1) equals to (30). Also, (2) on $x_{P}$ implies (32). The forward direction is thus proven.

We now prove that any solution in (30)–(32) leads to a valid solution in (1)–(2). We prove this by an iterative construction. Initially, we define $P = P_D$ and $x_{e}^{(h)} = x_{e}^{(h)}$ for all $e$ and $h$. For any $P \in P$, we choose $x_{P} = \min_{h \in [1, |P|]} x_{e}^{(h), P}$ where $e_{P, h}$ is the $h$-th hop of $P$. After choosing $x_{P}$, we decrease the value of $x_{e}^{(h), P}$ by $x_{P}$ for all $h \in [1, |P|]$ and remove $P$ from $P$. After decreasing the $x_{e}^{(h), P}$ values and reducing $P$, we repeat the construction for another $P \in P$ until $P = \emptyset$.

We now state and prove three claims regarding the above construction.

Claim 1: Throughout the process, all the $x_{e}^{(h)}$ are non-negative and they satisfy (31). The non-negativity holds since $x_{P} = \min_{h \in [1, |P|]} x_{e}^{(h), P}$ Equality (31) holds since we subtract the delay-constrained capacity upper bounds are provided, which for the single-unicast setting and the gain can be arbitrarily small in the end of each iteration, where $\{x_{P}\}$ are the latest values assigned to each path $P$, $\{x_{e}^{(h)}\}$ are the original LP variable values we begin with, and $\{x_{e}^{(h)}\}$ are the latest residual values in our construction. Then by (32) and by the non-negativity of $x_{e}^{(h)}$, the resulting $x_{P}$ in the end ($i.e.$, when $P = \emptyset$) must satisfy (2).

Claim 2: The expression (1) computed from the final $x_{P}$ equals (30) computed from $x_{e}^{(h)}$. To prove this claim, we notice that since $x_{P}$ is deduced from $x_{e}^{(h), P}$ during our construction, we always have

$$\forall e \in E, \sum_{P : P \ni e, P \in P_D \setminus P} x_{P} + \sum_{h=1}^{D} x_{e}^{(h)} = \sum_{h=1}^{D} x_{e}^{(h)} \quad \text{(33)}$$

in the end of each iteration, where $\{x_{P}\}$ are the latest values assigned to each path $P$, $\{x_{e}^{(h)}\}$ are the original LP variable values we begin with, and $\{x_{e}^{(h)}\}$ are the latest residual values in our construction. Then by (32) and by the non-negativity of $x_{e}^{(h)}$, the resulting $x_{P}$ in the end ($i.e.$, when $P = \emptyset$) must satisfy (2).

Claim 3: The expression (1) computed from the final $x_{P}$ equals (30) computed from $x_{e}^{(h)}$. To prove this claim, we notice that by the construction of $P_D$, the last edge of any $P \in P_D$ must belong to $\text{In}(d)$. Since (33) holds for any edge $e \in \text{In}(d)$, we only need to prove that $x_{e}^{(h)} = 0$ for all $e \in \text{In}(d)$ and $h \in [1, D]$ in the end of our construction.

We prove this by contradiction. Suppose not. Then we have $x_{e}^{(h)} > 0$ for some $e \in \text{In}(d)$ and $h$. Since $\{x_{e}^{(h)}\}$ satisfies (31) for all $v \in V \setminus \{s, d\}$ and $h$, we can find $h$ edges, denoted by $\bar{e}_1$ to $\bar{e}_{h}$, satisfying simultaneously (i) $\bar{e}_i : \bar{e}_{i+1}$; (ii) $\bar{e}_1 \bar{e}_2 \cdots \bar{e}_{h}$ form a path of length $\bar{h}$, which is denoted by $\bar{P}$; and (iii) $x_{\bar{e}_i}^{(h)} > 0$ for $i \in [1, h]$. Since $\bar{e}_i : \bar{e}_{i+1}$, we have $\text{head}(\bar{e}_i) = d$. We now prove $\text{tail}(\bar{e}_1) = s$ by contradiction. Suppose not. Then we focus on (31) with $h = 1$ and $v = \text{tail}(\bar{e}_1)$. One can see that the left-hand side of (31) is zero since $x_{\bar{e}_i}^{(0)} = 0$ in our convention but the right-hand side is no less than $x_{\bar{e}_i}^{(1)} > 0$.
This contradiction implies that tail($\hat{e}_1$) = s. As a result, $\hat{P}$ connects $s$ and $d$ using $\hat{h} \leq D$ hops. Therefore $\hat{P} \in \mathcal{P}_D$.

On the other hand, all the paths in $\mathcal{P}_D$ must have been considered in the iterative construction. Consequently, for any path $P \in \mathcal{P}_D$, $x_i^{(h)} = 0$ for at least one $h \in [1, |P|]$ since we subtract $x_P = \min_{h \in [1, |P|]} x_i^{(h)}$ from all $x_i^{(h)}$. Property (iii) of $\hat{P} \in \mathcal{P}_D$ thus contradicts the fact that we have exhaustively considered all $P \in \mathcal{P}_D$. Claim 3 is thus proven.

Jointly, Claims 1 to 3 complete the proof.

Since (30)–(32) is a polynomial-time computable version of the maximum flow problem (1)–(2), in this work we have derived the following polynomial-time computable version of the minimum cut problem (3)–(4).

**Corollary 2:** We can compute $R_{\text{route}}$ by the following LP problem with $|E|$ non-negative variables $y_v \geq 0$, $\forall v \in E$, and $(|V| - 2) \cdot D$ real-valued variables $y_v^{(h)}$, $\forall v \in V \setminus \{s, d\}, h \in [1, D]$ such that

$$\min_{y_v \geq 0, y_v^{(h)}} \sum_{v \in E} y_v c_{e}$$

s.t.

$$y_v + y_{\text{head}(e)}^{(h)} - y_{\text{tail}(e)}^{(h)} \geq 0, \forall e \in E, \forall h \in [1, D],$$

where in (35) we use the convention $y_v^{(0+1)} = 0$ for all $v \in V$ and $y_v^{(h)} = y_v^{(h)} = 0$ for all $h \in [1, D]$.

**Proof:** One can easily verify that Corollary 2 is the dual of Proposition 7.

---

**APPENDIX B**

**THE HIGH-LEVEL DESCRIPTION OF THE PROOF OF PROPOSITION 2**

The proof of Proposition 2 contains three major ingredients. Firstly, we will generalize the example in Section IV-B and describe a family of network instances $\{(G_m, D_m) : \forall m \in \mathbb{N}\}$ indexed by $m$ such that the corresponding NC gain, denoted by $\text{gain}_m$, is strictly larger than 1 for all $m \in \mathbb{N}$. Secondly, we will provide an iterative genie-aided construction such that with the help of a genie we can use a network instance of $(G, D)$ with NC gain to construct another network instance $(G', D')$ with NC gain’ such that gain’ > gain. Furthermore, when the iteration continues indefinitely the final gain’ approaches 2 asymptotically. The family of network instances $\{(G_m, D_m) : \forall m \in \mathbb{N}\}$, introduced as the first component, is essential in this iterative construction. Finally, we will describe how the iterative construction can be implemented without the help of a genie.

The family of network instances is described in Appendix C. The iterative genie-aided construction is described in Appendix D. Appendix E describes how to perform iterative construction without the help of a genie.

**APPENDIX C**

**A FAMILY OF NETWORK INSTANCES WITH gain > 1**

In this section, we generalize the result in Section IV-B and describe a set of network instances $\{(G_m, D_m) : \forall m \in \mathbb{N}\}$ such that the corresponding NC gain, denoted by $\text{gain}_m$, is strictly larger than 1.

---

**Fig. 6.** A high-level description of the network instance that was previously described in Fig. 2.

**A. A high-level network description**

To that end, we first represent the network in Fig. 2 by an equivalent but more high-level description in Fig. 6. The high-level description contains two components. The primary component is the two triangles (one on the left and one on the right) plus the straight line from $s$ to $d$. We use thick lines in Fig. 6 to represent the primary component. The two triangles symbolize the two possible detours from the straight line. One is to use node $u_1$ and the other is to use node $w_1$. Comparing Figs. 2 and 6, we can view node $u_1$ as a relabeling of node $v_5$ and node $w_1$ as a relabeling of node $v_7$. Note that each thick line in Fig. 6 does not correspond to an edge in Fig. 2. Instead, a thick line in Fig. 6 corresponds to a path in Fig. 2.

For example, since we view $u_1$ as a relabeling of node $v_5$, the thick line connecting $u_1$ and the tail of $e^*$ corresponds to the 2-edge path $v_5v_6v_2$ in Fig. 2 and the thick line connecting $s$ and $u_1$ corresponds to the 1-edge path $sv_5$ in Fig. 2. Therefore, the left detour from $s$ along $u_1$ to the tail of $e^*$ has length 3 even though it is represented by two thick line segments.

We choose the delay requirement $D = 6$ in a way that if we detour for exactly once, e.g., the longer path using $u_1$ but not using $w_1$, then the total length is exactly $D$. Equivalently, it means that the choice $D = 6$ implies that if we detour for two times, using both $u_1$ and $w_1$, then the resulting path will have length $> D$. For example, the path $sv_5v_6v_2v_3v_7v_8d$ in Fig. 2 corresponds to using both the detour through $u_1$ and the detour through $w_1$. Such path has 7 hops, which is $> D = 6$.

The secondary component of the high-level description in Fig. 6 is a parallel pipe connecting $(A_1, B_1)$ that does not use any edges in the primary component. We use a thin curve to represent this parallel pipe. We require that the delay incurred by the $(A_1, B_1)$ parallel pipe is identical to the delay when traversing from $A_1$ to $B_1$ using the edges in the primary component. Comparing Figs. 2 and 6, we can view node $A_1$ as a relabeling of node $v_1$ and node $B_1$ as a relabeling of node $v_4$ in Fig. 2. The parallel pipe in Fig. 2 thus corresponds to the path $v_1v_2v_3v_4$ in Fig. 2. The delay incurred by the parallel pipe is 3, which is equal to the delay when traversing through the path $v_1v_2v_3v_4$ in the primary component.

All edges in our high-level description in Fig. 6 are of capacity $c_e = 1$.

We can now explain the NC solution using the high-level description of Fig. 6. Consider sending $X_t$ through the primary component using the detour of $w_1$ only (i.e., path $sv_1v_2v_3v_7v_8d$ in Fig. 2). For easier reference, we term the above operation “sending $X_t$ through the ‘primary’ path $sv_1w_1d$” to emphasize that we only use the edges in the primary component and the notation “$u_0$” indicates that we do not use the detour corresponding to $u_1$. Similarly, we can...
send \(Y_t\) through the primary path \(su_1w_0d\), which emphasizes that we use the detour corresponding to \(u_1\) and do not use the detour corresponding to \(w_1\). Recall that the \(D\) value equals to the length of path \(su_0w_1d\) and equals to the length of path \(su_1w_0d\). Therefore, the intuition is that by sending \(X_t\) and \(Y_t\) along paths \(su_0w_1d\) and \(su_1w_0d\), respectively, our goal is for \(d\) to receive both \(X_t\) and \(Y_t\) before the deadline \(t+D\) through the two primary paths, respectively. For easier reference, we say that \(X_t\) (resp. \(Y_t\)) is the desired packet along path \(su_0w_1d\) (resp. \(su_1w_0d\)).

However, the two primary paths \(su_0w_1d\) and \(su_1w_0d\) share a bottleneck edge \(e^*\) in Fig. 6, or equivalently edge \(v_2v_3\) in Fig. 2. If we perform NC (i.e., simple packet addition) at \(e^*\) without using the \((A_1, B_1)\) parallel pipe, then by the deadline \(t+D\), \(d\) will receive packet \([X_t + Y_{t-\delta}]\) through path \(su_0w_1d\) for some \(\delta > 0\) that represents the difference of the distances between the detour through \(u_1\) and the direct path in the left triangle. Namely, the desired \(X_t\) packet will be corrupted by some old packet \(Y_{t-\delta}\) since \(Y_t\) needs to traverse a longer path (detour through \(u_1\)) before entering the bottleneck edge \(e^*\) while \(X_t\) traverses to \(e^*\) through a shorter direct path. Similarly, by the deadline \(t+D\), \(d\) will receive packet \([Y_t + X_{t+\delta}]\) through path \(su_1w_0d\). Namely, the desired \(Y_t\) packet will be corrupted by some future packet \(X_{t+\delta}\) since \(X_t\) traverses to \(e^*\) through a shorter direct path.

To prevent the desired \(Y_t\) (along the primary path \(su_1w_0d\)) from the corruption of the future packet \(X_{t+\delta}\), we use the \((A_1, B_1)\) parallel pipe to remove the aforementioned corruption. Recall that the \((A_1, B_1)\) parallel pipe incurs the same amount of delay as the primary path from \(A_1\) to \(B_1\). As a result, at node \(B_1\) we can subtract the corruption \(X_{t+\delta}\) (known from the information received through the parallel pipe) from the linear sum \([Y_t + X_{t+\delta}]\), which is received through the downstream path from \(e^*\) to \(B_1\) in the primary component. After subtraction, \(d\) will receive pure, uncorrupted \(Y_t\) through \(su_1w_0d\) before the deadline \(t+D\).

Note that there is no need to introduce additional parallel pipe for the task of subtracting the corruption \(Y_{t-\delta}\) for the desired \(X_t\) along the \(su_0w_1d\) path. The reason is that \(d\) has already decoded the old packet \(Y_{t-\delta}\) in the previous time slots. Therefore, \(d\) can simply use its own knowledge of \(Y_{t-\delta}\) to remove the corruption \(Y_{t-\delta}\). In summary, by the deadline \(t+D\), \(d\) can decode \(X_t\) from the \(su_0w_1d\) path (with the help of previously obtained knowledge of \(Y_{t-\delta}\)) and can receive pure uncorrupted \(Y_t\) from the \(su_1w_0d\) path (with the help of the \((A_1, B_1)\) parallel pipe).

In contrast with the edge-by-edge analysis in Section IV-B, this high-level description emphasizes the network topology and the corresponding network information flow, which will be useful when describing more complicated examples in the subsequent discussion.

B. A family of network instances

From the high-level description, one can see that the key ideas of Fig. 6 are (i) Create 2 primary paths of length equal to \(D\) (paths \(su_0w_1d\) and \(su_1w_0d\)); (ii) Make sure that the two paths use the bottleneck edge \(e^*\) in the center; (iii) Make each path correspond to “a route that detours in only one location”; (iv) Choose the \(D\) value such that detouring in two locations will violate the delay constraint; and (v) Finally use the parallel pipe to provide side information so that we can remove the corruption of “future interference \(X_{t+\delta}\)” from the path \(su_1w_0d\) so that \(d\) can receive uncorrupted \(Y_t\) by the deadline \(t+D\). In a broad sense, the use of the \((A_1, B_1)\) parallel pipe alleviates the bottleneck edge \(e^*\), which is used by both primary paths.

The following construction demonstrates how we can create \(m \geq 2\) primary paths of length equal to \(D\) and make sure all the \(m\) paths are using the same bottleneck edge \(e^*\) so that we can create more “congestion” in \(e^*\), which, heuristically, should lead to a higher NC gain. Again our construction consists of the primary and the secondary components.

1) The primary component: For any fixed integer \(m \geq 2\), the primary component of our construction has \(2(m-1)\) triangles; \((m-1)\) of them are nested on the left-hand side of the network; \((m-1)\) of them are nested on the right-hand side of the network; and a straight line from \(s\) to \(d\) that connects the two groups of triangles. See Fig. 7 for the illustration of \(m = 4\), in which the paths of the primary component are represented by thick lines.

We use the notation \(su_iz_jd\), where \(i, j \in [0, m-1]\), to denote the path that detours through \(u_i\) and \(w_j\) in the left and the right triangles, respectively. The notation \(w_0\) (resp. \(w_0\)) means no detour in the left half (resp. the right half) of the network. We call the path \(su_iz_jw_{m-1-i-d}\) the primary path. The sizes of the triangles are carefully chosen so that the primary paths \(su_iz_jw_{m-1-i-d}\), \(\forall i \in [0, m-1]\) have the same common length. Additionally, the sizes of the triangles are chosen in a way that the larger the triangle, the longer the corresponding detour will be. Take Fig. 7 for example. All the primary paths having the same length means

\[
\text{length}(su_0w_3d) = \text{length}(su_1w_2d) = \text{length}(su_2w_1d) = \text{length}(su_3w_0d).
\]  

(36)

The assumption “the larger the triangle, the longer the detour” means

\[
\text{length}(su_0w_0d) < \text{length}(su_1w_0d) < \text{length}(su_2w_0d) < \text{length}(su_3w_0d) \quad \text{and} \quad \text{length}(su_0w_3d) < \text{length}(su_0w_2d) < \text{length}(su_0w_3d).
\]  

(37)
Eqs. (36) and (37) together imply many other inequalities. For example, we also have
\[
\text{length}(su_1w_3d) > \text{length}(su_0w_3d) = \text{length}(su_2w_3d) > \text{length}(su_2w_0d).
\]
After fixing the network topology, we choose the delay requirement $D_m$ such that
\[
D_m = \text{length}(su_iw_{m-1-i}d), \quad \forall i = 0, \ldots, m - 1,
\]
is the length of all $m$ primary paths.

2) The secondary component: The secondary component contains $(m - 1)$ parallel node-disjoint pipes connecting $(A_i, B_i)$ for $i = 1$ to $m - 1$, which do not use any nodes/edges in the primary component except for the end nodes $A_i$ and $B_i$. We use thin curves to represent these parallel node-disjoint pipes. We now describe the starting and ending nodes of these $m - 1$ pipes.

Starting node $A_i$:
- If $i = 1$, then we choose $A_1$ to be a node satisfying simultaneously (i) $A_1$ is an interior node of the direct path $su_0w_0d$; (ii) $A_1$ is strictly upstream of the node where the direct path $su_0w_0d$ merges with the detour path $su_1w_0d$, i.e., $A_1$ is on the left of the merge node. See Fig. 7 for illustration.
- If $i \in [2, m-1]$, then we choose $A_i$ to be a node satisfying simultaneously (i) $A_i$ is strictly downstream of the node where the direct path $su_0w_0d$ merges with the detour path $su_{i-1}w_0d$, i.e., $A_i$ is on the right of the merge node; and (ii) $A_i$ is strictly upstream of the node where the direct path $su_0w_0d$ merges with the detour path $su_iw_0d$. See Fig. 7 for illustration.

Ending node $B_i$:
- If $i \in [1, m-2]$, then we choose $B_i$ to be a node satisfying simultaneously (i) $B_i$ is an interior node of the detour path $su_0w_{m-1-i}d$; and (ii) $B_i$ is strictly downstream of the node where the direct path $su_0w_0d$ diverges from the detour path $su_0w_{m-1-i}d$. See Fig. 7 for illustration.
- If $i = m - 1$, then we choose $B_{m-1}$ to be a node satisfying simultaneously (i) $B_{m-1}$ is an interior node of the direct path $su_0w_0d$; (ii) $B_{m-1}$ is strictly downstream of the node where the direct path $su_0w_0d$ diverges from the detour path $su_0w_1d$. See Fig. 7 for illustration.

One can clearly see that the choices of $A_i$ and $B_i$ are not unique. Our construction holds for any arbitrary choices satisfying the above description.

After fixing the end nodes $A_i$ and $B_i$, we impose that the delay incurred by the $(A_i, B_i)$ parallel pipe is identical to the delay when traversing from $A_i$ to $B_i$ using only the edges in the primary component. We use $(G_m, D_m)$ to describe the resulting network instance in the above construction. We assume that all edges in our construction (Fig. 7) are of capacity $c_e = 1$. Our construction is now complete.

Intuition: One way to interpret our construction is to view it (see Fig. 7) as an overlay of $m$ paths of length $D_m$ in an offset manner so that they form two groups of triangles (detours), one on the left and one on the right. The overlay is made so that all $m$ paths share a common bottleneck edge $e^*$. Then introduce parallel pipes that can potentially be used to remove the undesired corruption in the bottleneck edge $e^*$. Also see our discussion in Appendix C-A.

Subsequently, we will quantify the NC gain for the network instance $(G_m, D_m)$ for any given fixed integer $m \geq 2$.

C. Quantifying the NC gain of the proposed network family

Lemma 1: For any given $m \geq 2$, the delay-constrained NC capacity of $(G_m, D_m)$ is $R_{NC} = m$ packets per time slot.

Proof: We prove this lemma by explicit network code construction. For all $i = 0$ to $m - 1$, we send $X_i^{[i]}$ through the primary path $su_iw_{m-1-i}d$. With (38), we are hoping that $d$ can receive the desired $X_i^{[m]}$ path through path $su_iw_{m-1-i}d$ before the deadline $t + D_m$. See Fig. 7 for illustration.

Define the “merging nodes” as the nodes for which the $su_iw_0d$ path merges with the $su_{j+1}w_0d$ paths for all $j = 0$ to $m - 2$. If we perform NC (i.e., simple packet addition) on all the merging nodes while neglecting all parallel pipes in the secondary component, then by the deadline $t + D_m$ the packet received by $d$ through path $su_iw_{m-1-i}d$ will have the following form:
\[
\left(\sum_{j=0}^{i-1} X_j^{[i]} + t_{j;i}\right) + X_i^{[i]} + \sum_{j=i+1}^{m-1} X_j^{[i]} - t_{j;i} \Delta \] (39)
where
\[
|\delta_{j;i}| = |\text{length}(su_jw_0d) - \text{length}(su_{i}w_0d)|
\]
is the absolute value of the difference between the distance from $s$ to $e^*$ using detour $uj$ and the distance from $s$ to $e^*$ using detour $ui$. By our assumption in (37), for those $j < i$ the packet $X_j^{[i]}$ sent through the primary path $su_jw_{m-1-j}d$ will arrive at $e^*$ earlier than $X_i^{[i]}$ does. This is why when focusing on $X_i^{[i]}$ the corruption for those $j < i$ is of the form $X_j^{[i]}$. Namely, those $j$ values correspond to corruption from the “future packets.” Similarly, for those $j > i$ the packet $X_j^{[i]}$ sent through the primary path $su_jw_{m-1-j}d$ will arrive at $e^*$ later than $X_i^{[i]}$ does. The corruption is thus of the form $X_j^{[i]}$, which corresponds to the corruption from the “old packets.”

We now discuss how to use the $\{(A_i, B_i) : i \in [1, m-1]\}$ parallel pipes. Again, we consider a fixed $i$ value and the packet received by $d$ through path $su_iw_{m-1-i}d$. Without loss of generality, we assume $i \geq 1$ and the scenario of $i = 0$ is a degenerate case. Recall that by our construction $A_i$ is located in paths $su_jw_{m-1-j}d$ for all $j \in [0, i-1]$ but not in the path of interest $su_iw_{m-1-i}d$. Namely, the location of $A_i$ allows node $A_i$ to observe the interference term $\left(\sum_{j=0}^{i-1} X_j^{[i]} + t_{j;i}\right)$ before the interference corrupts the desired $X_i^{[i]}$ term.

Also recall that by our construction $B_i$ is located in the path $su_iw_{m-1-i}$ and the $(A_i, B_i)$ parallel pipe in the secondary component incurs the same amount of delay as the path from $A_i$ to $B_i$ in the primary component. As a result, node $A_i$ can transmit the interference term $\left(\sum_{j=0}^{i-1} X_j^{[i]} + t_{j;i}\right)$ to node $B_i$ through the parallel pipe and node $B_i$ can use this information
to subtract the corruption from the linear sum in (39). After the subtraction performed by \( B_i \), \( d \) will receive

\[
X_t^{[i]} + \left( \sum_{j=i+1}^{m-1} X_t^{[j]} \right).
\]

through the primary path \( su_iw_{m-1-i}d \) by the deadline \( t + D_m \). Since the remaining corruption term \( \sum_{j=i+1}^{m-1} x_t^{[j]} \) is resulted from the “old packets”, \( d \) can use its existing knowledge about the previously decoded packets to subtract the corruption and decode the desired \( X_t^{[i]} \) along the primary path \( su_iw_{m-1-i}d \). Since \( d \) can decode \( X_t^{[i]} \) through \( su_iw_{m-1-i}d \) for all \( i \in [0, m-1] \), the delay-constrained NC throughput is \( m \) packets per time slot. Since the min-cut value from \( s \) to \( d \) is also \( m \), the above scheme is throughput optimal and the delay-constrained NC capacity \( R_{NC} \) is indeed \( m \) packets per time slot.

We now quantify the routing-based capacity \( R_{route} \).

**Lemma 2**: For any given \( m \geq 2 \), the delay-constrained routing capacity of \( (G_m, D_m) \) is \( R_{route} = m-1+2^{-m} \) packets per time slot.

**Proof**: We prove this lemma by providing a primal and a dual solutions for (1)–(2) and for (3)–(4), respectively, and then show that the duality gap is zero.

For the primal variables \( x_P \), we define \( 2m-1 \) paths, denoted by \( \{P_i, i \in [0, m-1]\} \) and \( \{Q_j : j \in [1, m-1]\} \), respectively. They are

\[
P_i \triangleq su_iw_{m-1-i}d \quad \text{and} \quad Q_j \triangleq su_{j-1}A_jB_jw_{m-1-j}d.
\]

Namely, \( P_i \) is one of the primary paths. Path \( Q_j \) is a path that uses the \( (A_j, B_j) \) pipe in the secondary component and before using \( (A_j, B_j) \) the path \( Q_j \) uses the detour corresponding to \( u_{j-1} \). Then we set the primal variables as

\[
x_P = 2^{-m} - 1,
\]

\[
i \in \{1, \ldots, m-1\} \quad \text{and} \quad x_P = 2^{-m} - 1,
\]

\[
j \in \{1, \ldots, m-1\} \quad \text{and} \quad x_{Q_j} = 1 - 2^{-m-j},
\]

and all other \( x_P = 0 \).

One can verify that the above \( x_P \) assignment is a feasible primal solution with the objective value being \( \sum_{P \in P_{dom}} x_P = m-1+2^{-m-1} \).

For the dual variables \( y_e \), we consider \( 2m-1 \) edges: For all \( i \in [1, m-1] \), we define \( e_{A_i} \) as the unique edge satisfying head\((e_{A_i}) = A_i \) and \( e_{B_i} \) as the unique edge satisfying tail\((e_{B_i}) = B_i \). Namely, \( e_{A_i} \) is the edge that is the direct upstream of node \( A_i \) and \( e_{B_i} \) is the direct downstream of node \( B_i \). Recall that \( e^* \) is the bottleneck edge in the center. Then we set the dual variables as

\[
y_e = 2^{-m} - 1,
\]

\[
i \in \{1, \ldots, m-1\} \quad \text{and} \quad y_{e_{A_i}} = 2^{-i}, \quad y_{e_{B_i}} = 1 - 2^{-i},
\]

and all other \( y_e = 0 \).

One can verify that the above \( y_e \) assignment is a feasible dual solution with the objective value being \( \sum_e y_e e_e = m-1+2^{-m-1} \). The proof is thus complete.

In sum, the NC gain \( m \) is

\[
\text{gain}_m = \frac{m}{m-1+2^{-m-1}}
\]

for the \( m \)-th network instance \( (G_m, D_m) \). One can easily verify that \( \text{gain}_m \leq 1 \) for large \( m \) and it attains this bound for \( m \) small enough.

**Remark**: From the above construction should yield high NC gain since all \( m \) primary paths \( su_iw_{m-1-i}d \) use the same bottleneck edge \( e^* \). However, detailed analysis in Lemmas 1 and 2 shows that even if we create a highly-congested bottleneck \( e^* \), the NC gain does not increase any further and is still upper bounded by \( \frac{4}{3} \). The reason is that in the proposed optimal NC solution, the parallel pipes are used to carry the side information that will be used for subtracting the “corruption caused by the future packets”. However, for a routing solution, we can directly use those parallel pipes to carry the uncoded information, see (41) and (42). As a result, simply increasing the congestion level at \( e^* \) does not lead to a network instance with a larger NC gain, and the simplest example with \( m = 2 \) still admits the largest NC gain \( \frac{4}{3} \) we have found thus far. In the next sections, we will demonstrate how to take advantage of the special structure of \( (G_m, D_m) \) for large \( m \) and use it to design a network instance with NC gain arbitrarily close to 2.

### Appendix D

**A genie-aided iterative construction**

In this section, we will present the main principles of our iterative construction, which will be based on the new concepts of throughput/delay (T/D) spectrum and T/D spread.

**A. The Throughput/Delay (T/D) spectrum and spread**

Given any network instance, the delay-constrained NC capacity can be written as a function \( R_{NC}(D) \) of the delay requirement \( D \). Here we use the calligraphic \( \mathcal{R} \) to emphasize that it is a function of \( D \). Obviously \( R_{NC}(D) \) is non-decreasing with respect to \( D \). Similarly, the delay-constrained routing capacity can be defined as \( R_{route}(D) \). We call this pair of functions \( (R_{NC}(D), R_{route}(D)) \) the throughput/delay (T/D) spectrum of the network. For example, by simple computation one can show that the network in Fig. 2 has

\[
\mathcal{R}_{NC}(D) = \begin{cases} 
0 & \text{if } D \leq 4 \\
1 & \text{if } D = 5 \\
2 & \text{if } D \geq 6 
\end{cases}
\]

\[
\mathcal{R}_{route}(D) = \begin{cases} 
0 & \text{if } D \leq 4 \\
1 & \text{if } D = 5 \\
1.5 & \text{if } D = 6 \\
2 & \text{if } D \geq 7 
\end{cases}
\]

Although the T/D spectrum \( (\mathcal{R}_{NC}(D), \mathcal{R}_{route}(D)) \) is well-defined, it may be difficult to compute for a general network topology. On the other hand, the T/D spectrum \( (\mathcal{R}_{NC}(D), \mathcal{R}_{route}(D)) \) can be used to compute a simpler concept called the T/D spread.
**Definition 5:** For any network with T/D spectrum \((R_{NC}(D), R_{route}(D))\), the T/D spread around a given D is a tuple \((\text{gain}, \Delta_L, \Delta_H)\), where

\[
\Delta_L \triangleq \sup\{x \in \mathbb{N} : R_{NC}(D-x) > 0\};
\]

and

\[
\Delta_H \triangleq \inf\{x \in \mathbb{N} : R_{route}(D-x) > R_{route}(D)\}.
\]

Namely, whenever the delay requirement satisfies \(D' < D - \Delta_L\), we will have \(R_{NC}(D') = R_{route}(D') = 0\). Whenever the delay requirement satisfies \(D \leq D' < D + \Delta_H\), we have \(R_{route}(D') = R_{route}(D)\). Intuitively, \(\Delta_L\) describes the lower spread around \(D\) before the throughput \(R_{NC}(D')\) and \(R_{route}(D')\) drop completely to zero, and \(\Delta_H\) describes the upper spread before the routing-based throughput \(R_{route}(D')\) increases.

For example, with \(D = 6\) the network in Fig. 2 has \((\text{gain}, \Delta_L, \Delta_H)_{D=6} = (\frac{4}{3}, 1, 1)\). We sometimes slightly abuse the notation and refer the pair \((\Delta_L, \Delta_H)\) as the T/D spread. It should be clear from the context whether the term T/D spread is referring to a tuple or a pair.

**B. Illustration of a genie-aided construction**

The example in Fig. 2 has T/D spread being \((\text{gain}, \Delta_L, \Delta_H)_{D=6} = (\frac{4}{3}, 1, 1)\). For easier reference, we denote the network in Fig. 2 by \(\tilde{G}\). In this subsection, we assume that there is a genie that can convert the network \(\tilde{G}\) to another finite network \(G^o\) such that \(G^o\) has T/D spread being \((\text{gain}^o, \Delta_L^o, \Delta_H^o)_{D^o} = (\frac{4}{3}, 0, \infty)\) around a new delay constraint \(D^o\) that may be different from \(D = 6\). Namely, the resulting \(G^o\) has the same NC gain as the original \(\tilde{G}\) but has a different T/D spread. In general, such a conversion is impossible since one can prove that any network instance \((G, D)\) with gain \(> 1\) must have \(\Delta_L > 0\) and \(\Delta_H < \infty\). However, for sake of discussion, we assume such a genie exists and call the resulting network \(G^o\) a fictitious network to distinguish it from an actual network instance.

We now demonstrate how the fictitious network \(G^o\) can be used to construct a network instance of \((G, D)\) with gain \(= \frac{16}{11}\). Without loss of generality we assume that \(R_{NC}(D^o) = 1\) and \(R_{route}(D^o) = \frac{2}{3}\) for the fictitious network \(G^o\) since the corresponding gain is \(\frac{4}{3}\). This can be achieved by scaling the capacity of each edge of \(G^o\) proportionally until \(R_{NC}(D^o) = 1\) and \(R_{route}(D^o) = \frac{2}{3}\). Our construction will combine the original network in Fig. 2 with the new fictitious network \(G^o\).

Specifically, we replace each of the two edges \((v_4, d)\) and \((v_8, d)\) of Fig. 2 by a copy of the \(G^o\) network, respectively. The description of the topology of the new network \(G^o\) is now complete and the resulting graph is illustrated in Fig. 8(a).

Recall that the NC gain of \(G^o\) is \(\frac{4}{3}\) at some fixed delay constraint \(D^o\). For the new network in Fig. 8(a), we set the new delay requirement to be \(D = 5 + D^o\). The description of the network instance \((G, D)\) is now complete. What remains

\[15\text{In Appendix E, we will provide a construction that directly uses } \tilde{G}, \text{ without using the fictitious network } G^o.\]

\[16\text{Our construction treats } G^o \text{ as a black box and does not depend on the actual topology of } G^o.\]

```plaintext
Fig. 8. Three closely related network instances with the same delay requirement \(D = 5 + D^o\). The fictitious network \(G^o\) has T/D spread \((\text{gain}^o, \Delta_L^o, \Delta_H^o) = (\frac{4}{3}, 0, \infty)\).
```

to prove is that the new \(G\) in Fig. 8(a) has gain \(= \frac{16}{11}\) when \(D = 5 + D^o\).

We now argue that the delay-constrained NC capacity is still 2 packets per time slot. The reason is that since \(R_{NC}(D^o) = 1\), we can always send 1 packet per time slot through the input terminal of \(G^o\), perform NC in the interior of \(G^o\), and extract the information packets from the output terminal of \(G^o\) after \(D^o\) slots. Therefore, from a NC's perspective, the network in Fig. 8(a) is no different than replacing the \((v_4, d)\) and \((v_8, d)\) edges by two paths of capacity 1 and length \(D^o\), respectively. See Fig. 8(b). Since we set the new delay requirement to be \(D = 5 + D^o\), we can use the same analysis as in Section IV-B to prove that the NC capacity is 2 packets per time slot. The subtle difference herein is that the achievable scheme for Fig. 2 simply forwards the coded packets along the \((v_4, d)\) and \((v_8, d)\) edges. The achievable scheme for Fig. 8(a) needs to perform the optimal NC solution associated with \(G^o\) when sending packets along the copies of \(G^o\) that connects \(v_4\) and \(v_8\) to the destination \(d\).

We now argue that the delay-constrained routing capacity of Fig. 8(a) is \(\frac{1}{7}\) packets per time. To that end, the following Lemma 3 first proves that the network in Fig. 8(a) has the same delay-constrained routing capacity as the network in Fig. 8(c), which replaces the \((v_4, d)\) and \((v_8, d)\) edges by paths of capacity \(\frac{2}{3}\) and length \(D^o\). Using Lemma 3 we can then compute the delay-constrained routing capacity of Fig. 8(a) by applying the LP computation (1)–(2) to Fig. 8(c). The end
result shows that \( R_{\text{route}}^* = \frac{11}{8} \) for both Figs. 8(a) and 8(c). The overall NC gain for the network in Fig. 8(a) is thus gain = \( \frac{11}{14} = \frac{16}{14} \).

**Lemma 3:** The networks in Figs. 8(a) and 8(c) have the same routing-based capacity \( R_{\text{route}}^* \).

**Proof:** For easier reference, we call the network in Fig. 8(a) a \( G^o \)-compound network and the network in Fig. 8(c) a routing-equivalent network.

It is easy to see that the delay-constrained routing capacity of the routing-equivalent network (Fig. 8(c)) is always no larger than that of the \( G^o \)-compound network (Fig. 8(a)). The reason is that for whatever routing solution of the routing-equivalent network, we can always adapt it and find a routing solution for the \( G^o \)-compound network since by our construction the constituent subnetwork \( G^o \) is capable of supporting routing rate \( \frac{3}{4} \) within a hard delay requirement \( D^o \).

We now argue that the delay-constrained routing capacity of the \( G^o \)-compound network (Fig. 8(a)) is no larger than that of the routing-equivalent network (Fig. 8(c)). This direction is non-trivial since we do not know the underlying topology of the subnetwork \( G^o \) and the proof needs to hold for any \( G^o \) with T/D spread \( (\text{gain}^o, \Delta_H^o, \Delta_H^o) = \left( \frac{4}{3}, 0, \infty \right) \). Our proof consists of the following three observations.

**Observation 1:** Any path \( \tilde{P} \) in the \( G^o \)-compound network can be mapped to a path \( P \) in the routing-equivalent network by tracing the corresponding routes.

**Observation 2:** For any path \( \tilde{P} \) in the \( G^o \)-compound network, denote its image path in the routing-equivalent network by \( P \). If \( \text{length}(P) > D \), then we must also have \( \text{length}(\tilde{P}) > D \).

**Observation 3:** Any deadline-respecting path in the \( G^o \)-compound network must go through one and only one \( G^o \) network. Furthermore, the corresponding sub-path inside \( G^o \) must have length \( < (D^o + \Delta_H^o) \).

The proof of this observation is almost self-explanatory since (i) The collection of the two \( G^o \) in Fig. 8(a) form a cut; (ii) The two \( G^o \) cannot reach each other; and (iii) In this proof we assume the fictitious network \( G^o \) has \( \Delta_H^o = \infty \). However, the importance of this observation is significant. As will be proven in the next paragraph, Observation 3 implies that from a routing perspective, each of the two \( G^o \) subnetworks in the compound network is no different than a pipe of length \( D^o \) that supports rate \( \mathcal{R}_{\text{route}}(D^o) \).

The reason is as follows. Consider any arbitrary \( G^o \) subnetwork in the compound network. We are interested in the deadline-respecting paths (those having length \( \leq D = 5 + D^o \)) that go through the \( G^o \) of interest. Observation 3 implies that each of those deadline-respecting paths in the compound network uses a sub-path of length \( \leq D^o + \Delta_H^o - 1 \) in \( G^o \). Therefore, the sum of all rates assigned to those deadline-respecting paths must be upper bounded by \( \mathcal{R}_{\text{route}}(D^o + \Delta_H^o - 1) \), the largest supportable rate when performing routing over sub-paths of length \( \leq D^o + \Delta_H^o - 1 \) in \( G^o \). At the same time, our T/D spread definition we have \( \mathcal{R}_{\text{route}}(D^o) = \mathcal{R}_{\text{route}}(D^o + \Delta_H^o - 1) \). Therefore, the sum-rate of all deadline-respecting paths in the overall compound network that use \( G^o \) must be upper bounded by \( \mathcal{R}_{\text{route}}(D^o) \). The \( G^o \) subnetwork is thus no different than a pipe of length \( D^o \) and capacity \( \mathcal{R}_{\text{route}}(D^o) \) from the routing’s perspective.

With Observations 1 to 3, we are ready to complete the proof that the delay-constrained routing capacity of the \( G^o \)-compound network (Fig. 8(a)) is no larger than the delay-constrained capacity of the routing-equivalent network (Fig. 8(c)).

Denote the two \( G^o \) subnetworks in the \( G^o \)-compound network (Fig. 8(a)) by \( G_k^o \), where \( k = 1 \) or 2 depending on which \( G^o \) we choose. For any LP solution \( \{x_P : \tilde{P} \in \mathcal{P}_D\} \) of the
where \( P(G^o_k) \) denotes the special sub-path in the routing-equivalent network (Fig. 8(c)) that corresponds to the \( G^o_k \) sub-network of interest, and \( c_{P(G^o_k)} \) is the capacity of the path \( P(G^o_k) \) in the routing-equivalent network, which equals \( c_{P(G^o_k)} = \frac{1}{g_m} = \frac{3}{4} \). The equality in (46) follows from Observation 2 and (45). The inequality in (47) follows from our discussion of Observation 3, i.e., the sum of routing rates of all paths using \( G^o_k \) is upper bounded by \( R_{route}(D^o) \).

In addition to (46)–(47), the LP solution \( \{x_{\bar{P}} : \bar{P} \in \tilde{P}_D \} \) of the compound network (Fig. 8(a)) also satisfies that for any \( e \) that is not inside any \( G^o_k \), we have

\[
c_{e} = 1 \geq \sum_{\bar{P} \in \tilde{P}_D} x_{\bar{P}} = \sum_{i : P_i \text{ uses } e} \left( \sum_{\bar{P} \in P_i} x_{\bar{P}} \right)
\]  

(48)

where \( c_e = 1 \) follows from our compound-network construction; the inequality is the edge capacity constraint for any routing solution of the compound network; and the final equality follows from (45). Furthermore, the overall delay-constrained routing capacity of the compound network can be written as

\[
\max \sum_{\bar{P} \in \tilde{P}_D} x_{\bar{P}} = \max \sum_{i \in [P_0]} \left( \sum_{\bar{P} \in P_i} x_{\bar{P}} \right)
\]  

(49)

where the equality follows from (45).

We now prove that any feasible LP solution \( \{x_{\bar{P}} : \bar{P} \in \tilde{P}_D \} \) for the \( G^o \)-compound network can be used to construct another feasible LP solution \( \{x_{\bar{P}}^{[\text{pure}]} : i = 1, \ldots, |\tilde{P}_D| \} \) for the routing-equivalent network, and the resulting \( \{x_{\bar{P}}^{[\text{pure}]} \} \) has the same objective value as the original \( \{x_{\bar{P}} \} \). To that end, we simply set each \( x_{\bar{P}}^{[\text{pure}]} \equiv \sum_{\bar{P} \in \bar{P}_i} x_{\bar{P}} \) for all \( i \). The resulting \( \{x_{\bar{P}}^{[\text{pure}]} \} \) is feasible since (46) and (48) for the original \( \{x_{\bar{P}} \} \) ensure that the new \( \{x_{\bar{P}}^{[\text{pure}]} \} \) will satisfy the edge-capacity constraints for the routing-equivalent network. The new construction \( \{x_{\bar{P}}^{[\text{pure}]} \} \) also has the same objective value as that of \( \{x_{\bar{P}} \} \), since the objective (49) for the compound network can be transcribed as the objective for the routing-equivalent network.

The above argument shows that any feasible routing solution \( \{x_{\bar{P}} \} \) of the \( G^o \)-compound network can be used to construct another feasible routing solution \( \{x_{\bar{P}}^{[\text{pure}]} \} \) of the routing-equivalent network with the end-to-end throughput. As a result, we have proven that the delay-constrained routing capacity of the compound network (Fig. 8(a)) is no larger than that of the routing-equivalent network (Fig. 8(c)).

C. An iterative genie-aided construction of network instances with \( \text{gain} > 2 - \epsilon \)

Thus far, we have demonstrated how to find a network instance with \( \text{gain} = \frac{16}{11} \) when assuming there is a genie who can convert Fig. 2 with \( (\text{gain}, \Delta_L, \delta_H) = (\frac{1}{2}, 1, 1) \) to a fictitious network \( G^o \) with \( (\text{gain}^o, \Delta^o_L, \delta^o_H)_{D^o} = (\frac{4}{3}, 0, \infty) \). The resulting graph is depicted in Fig. 8(a).

Suppose the same genie is very powerful and can convert any given network with T/D spread \( (\text{gain}, \Delta_L, \delta_H) \) to a fictitious network \( G^o \) with \( (\text{gain}^o, \Delta^o_L, \delta^o_H)_{D^o} = (\text{gain}, 0, \infty) \), i.e., keeping the same gain but with the new \( \Delta^o_L = 0 \) and new \( \Delta^o_H = \infty \). Then we can iteratively use the construction in Appendix D-B to further improve the NC gain.

That is, we use the genie to convert the network in Fig. 8(a) to another fictitious network \( G^o \) with \( (\text{gain}^o, \Delta^o_L, \delta^o_H)_{D^o} = (\frac{11}{16}, 0, \infty) \). Without loss of generality, we assume the NC capacity and the routing capacity of the new \( G^o \) are \( R_{NC}(D^o) = 1 \) and \( R_{route}(D^o) = \frac{11}{16} \), respectively, since \( \text{gain}^o = \frac{16}{11} \). Then we use the new \( G^o \) and plug it into Fig. 8(a) again. By the same argument as used in Appendix D-B, the NC capacity of the new Fig. 8(a) (with the new \( G^o \)) is still 2 packets per slot. Also, by the same argument as used in the proof of Lemma 3, the routing capacity of the new Fig. 8(a) is equal to the delay-constrained routing capacity of Fig. 9.

Using the LP formulation (1)–(2), one can prove that the routing capacity of Fig. 9 is \( \frac{41}{32} \). The new NC gain over routing thus becomes \( \text{gain} = \frac{27}{40} > \frac{25}{32} = \frac{41}{32} \). We can then repeat the above process to continuously improve the NC gain.

By similar analysis, one can prove that if we start from any network with \( \text{gain}_\text{old} \leq 1.5 \), then after converting it to a fictitious network \( G^o \) with T/D spread \( (\text{gain}^o, \Delta^o_L, \delta^o_H)_{D^o} = (\text{gain}_\text{old}, 0, \infty) \) and using it to construct Fig. 8(a), the new network has NC gain being

\[
\text{gain}_\text{new} = \frac{2 \cdot \text{gain}_\text{old}}{0.5 + \text{gain}_\text{old}}.
\]  

(50)

Since we assume that we have \( \text{gain}_\text{old} \leq 1.5 \) to begin with, one can prove that \( \text{gain}_\text{new} \leq 1.5 \) as well. Furthermore, by solving the fixed point equation of (50), one can prove that the above construction can generate network instances with NC gains arbitrarily close to 1.5 after a sufficiently large number of iterations. Although \( \text{gain} = 1.5 \) is 12.5% improvement over the original gain \( \frac{16}{11} \), the gain is still strictly bounded away from 2 when using the above iterative construction.

In the following, we demonstrate how to modify the above procedure and generate network instances with NC gains \( \geq 2 - \epsilon \) for any arbitrarily given \( \epsilon > 0 \).

**Step 1:** For any given \( \epsilon > 0 \), we find an \( m \) value such that \( 2^{-m-1} < \epsilon \) and fix that \( m \) value throughout our construction.

**Step 2:** For any arbitrarily given network with NC gain satisfying \( 1 \leq \text{gain}_\text{old} \leq 2 - 2^{-m-1} \), we use the genie to...
Described in Step 5, the description of the new network topology described in Step 4 and the delay requirement value \(D\) the delay-constrained routing capacity of \(G\) delay-constrained NC capacity of \(G\) (Step 2 and the subsequent Steps 3 to 5. By iteratively repeating and generate another fictitious network to begin with, one can prove that Since we assume that we have \(m\) that enter the destination \(d\) when \(G; D\) needs to perform the optimal NC solution associated with \(G^o\) when sending packets along the copies of \(G^o\) that enter the destination \(d\).

We now prove that the new \((G, D)\) has delay-constrained routing capacity being

\[
R_{\text{route}}^* = \left( \frac{m - 2 + 2^{-(m-1)}}{\text{gain}_{\text{old}}} + 1 \right) \text{ packets per slot. (52)}
\]

To that end, we first notice that all arguments in the proof of Lemma 3 can be applied verbatim to our new construction. Therefore, the routing capacity of the new Fig. 10(a) is equal to the delay-constrained routing capacity of Fig. 10(b), where Fig. 10(b) is the corresponding routing-equivalent network that replaces the \(G^o\) subnetworks in Fig. 10(a) by paths of length \(D^o\) and capacity \(\frac{1}{\text{gain}_{\text{old}}}\). To compute the routing capacity of Fig. 10(b), we follow the same approach as used in the proof of Lemma 2 by presenting a primal and a dual solution with zero duality gap.

For the primal variables \(x_P\), we define \(2m - 1\) paths, denoted by \(\{P_i, i \in [0, m - 1]\}\) and \(\{Q_j : j \in [1, m - 1]\}\), in the same way as in (40) and (41). Then we set the primal variables to be

\[
x_{P_0} = 1 - \frac{1}{\text{gain}_{\text{old}}} (1 - 2^{-m-1}),
\]

\[
\forall i \in \{1, \cdots, m - 1\}, \ x_{P_i} = \frac{1}{\text{gain}_{\text{old}}} 2^{-m-i},
\]

\[
\forall j \in \{1, \cdots, m - 1\}, \ x_{Q_j} = 1 - \frac{1}{\text{gain}_{\text{old}}} (1 - 2^{-m-j}),
\]

and all other \(x_P = 0\).

One can verify that with the assumption of \(\text{gain}_{\text{old}} \leq 2 - 2^{-m-1}\), the above \(\{x_P\}\) assignment is a feasible primal solution with the objective value being \(\sum_P x_P = m - 2 + 2^{-(m-1)} + 1\). For the dual variables \(y_e\), we consider \(2m - 1\) edges: For all \(i \in [1, m - 1]\), we define \(e_{A_i}\) as the unique edge satisfying \(\text{head}(e_{A_i}) = A_i\) and \(e_{B_i}\) as the edge in \(\text{In}(d)\) that is downstream of \(B_i\). Namely, \(e_{A_i}\) is the edge that is the direct upstream of node \(A_i\) and has \(c_{e_{A_i}} = 1\), and \(e_{B_i}\) is the edge that has capacity \(c = \frac{1}{\text{gain}_{\text{old}}}\) and is a downstream edge of node \(B_i\). Recall that \(e^*\) is the bottleneck edge in the center.
Then we set the dual variables to be
\[ y_{e^*} = 2^{-(m-1)}, \]
\[ \forall i \in \{1, \cdots, m-1\}, \quad y_{e_{A_i}} = 2^{-i}, \quad y_{e_{B_i}} = 1 - 2^{-i}, \]
and all other \( y_e = 0. \)

One can verify that the above \( \{y_e\} \) assignment is a feasible dual solution with the objective value being
\[ \sum_e y_e c_e = \left( y_{e^*} + \sum_{i=1}^{m-1} y_{e_{A_i}} \right) \cdot 1 + \left( \sum_{i=1}^{m-1} y_{e_{B_i}} \right) \cdot \frac{1}{\text{gain}_\text{old}} = 1 + \frac{m - 2 + 2^{-(m-1)}}{\text{gain}_\text{old}}. \]

The proof is thus complete.

In summary, we have provided a genie-aided iterative construction that leads to network instances with NC gain arbitrarily close to 2.

**APPENDIX E**

**AN ITERATIVE CONSTRUCTION WITHOUT USING ANY GENIE**

In this section, we first demonstrate our proposed construction on the simplest example, which is in parallel with our discussion in Appendix D-B. Then we describe our proposed construction for the most general setting, which is in parallel with our discussion in Appendix D-C.

**A. An iterative construction without using any fictitious networks**

In this subsection, we will describe a compound network construction that uses any arbitrarily given network instance \((G^o, D^o)\) with \((\text{gain}^o, \Delta_L^o, \Delta_H^o)_D^o = \left( \frac{1}{2}, 1, 1 \right)\) to construct a network instance with \( \text{gain} = \frac{16}{17} \). Such a construction is in parallel to the discussion in Appendix D-B except that our construction is without the aid of a genie. Since our construction can be easily generalized to any \( 1 \leq \text{gain}^o \leq 1.5, \Delta_L^o > 0, \text{ and } \Delta_H^o < \infty \), we will simply use the tuple \((\text{gain}^o, \Delta_L^o, \Delta_H^o)_D^o\) when describing our scheme, instead of the actual numbers \((\frac{1}{2}, 1, 1)\). Without loss of generality, we also assume that for this \( G^o \) we have \( R_{\text{NC}}(D^o) = 1 \) and \( R_{\text{route}}(D^o) = \frac{1}{\text{gain}^o}. \)

Our construction needs the following definition.

**Definition 6:** For any given graph \( G \) and any positive integer \( \alpha \), the \( \alpha \)-elongated version of \( G \), denoted by \( \alpha \cdot G \) or simply \( \alpha G \) for brevity, is obtained by replacing every edge in \( G \) by a path of length \( \alpha \). The capacity of the new path in \( \alpha G \) is set to be the same as the capacity \( c_e \) of the edge \( e \) in \( G \) it replaces.

**Remark:** It is self-explanatory that if the original graph \( G \) has the T/D spread being \((\text{gain}, \Delta_L, \Delta_H)_D \) at some \( D \), then the new graph \( \alpha G \) has the T/D spread being \((\text{gain}, \alpha \Delta_L, \alpha \Delta_H)_D \) at a new delay point \( \alpha D \).

Given any \((G^o, D^o)\) with \((\text{gain}^o, \Delta_L^o, \Delta_H^o)_D^o\), we start from the high-level description in Fig. 6. For any given four strictly positive integer values \( \alpha_1, \alpha_0, \delta_1, \text{ and } \text{Right.Half} \), we can modify Fig. 6 and construct the network in Fig. 11(a). Specifically, the length of the detour using \( u_1 \) (the left triangle) is set to \( 2 + \delta_1 \). The length of the detour using \( w_1 \) (the right triangle) is set to \( \text{Right.Half} + \delta_1 \). The delay requirement value is set to \( D \triangleq 2 + 1 + \text{Right.Half} + \delta_1 \). One can easily check that such a \( D \) value will ensure that any deadline-respecting path can detour at most once. We also require that the last \( D_1 \triangleq \alpha_1 \cdot D^o \) unit-capacity edges of the route from \( w_1 \) to \( d \) is replaced by a path with length \( D_1 \) and capacity \( \frac{1}{\text{gain}^o} \). Similarly, the last \( D_0 \triangleq \alpha_0 \cdot D^o \) unit-capacity edges of the route from \( w_0 \) to \( d \) is replaced by a path with length \( D_0 \) and capacity \( \frac{1}{\text{gain}^o} \). See Fig. 11(a) for detailed illustration.

We now describe how to choose the four parameter values \( \alpha_1, \alpha_0, \delta_1, \text{ and } \text{Right.Half} \). In particular, we require them to satisfy the following conditions, for which the integer constants \( D \triangleq 2 + 1 + \text{Right.Half} + \delta_1, D_0 \triangleq \alpha_0 D^o, \text{ and } D_1 \triangleq \alpha_1 D^o \) have been defined previously.

- **The Feasibility Condition:**
  \[ \text{Right.Half} + \delta_1 \geq D_1 + 1 \quad (53) \]
  \[ \text{Right.Half} \geq D_0 + 2. \quad (54) \]

- **Condition 2:**
  \[ (2 + \delta_1) + 1 + (\text{Right.Half} + \delta_1) - D_1 
  + (D_1 - \alpha_1 \Delta_H^o) > D. \quad (55) \]

- **Condition 3:**
  \[ D - (2 + 1 + (\text{Right.Half} + \delta_1) - D_1) 
  < D_1 + \alpha_1 \Delta_H^o \]
  \[ D - (2 + 1 + \text{Right.Half} - D_0) < D_0 + \alpha_0 \Delta_H^o. \quad (57) \]

The feasibility conditions in (53)–(54) are to ensure that the path length values in Fig. 11(a) are consistent. Specifically, the detour using \( w_1 \) has length \( \text{Right.Half} + \delta_1 \). We need it to be
strictly larger than $D_1$ so that after replacing the last $D_1$ unit-capacity edges of the route from $w_1$ to $d$ by the special path with length $D_1$ and capacity $\frac{1}{\gamma}$, we still have part of the $w_1$-detour being the regular unit-capacity edges (the thick lines in Fig. 11(a)). Similarly, when not detouring in the right triangle, we need to have the direct path length $\text{Right.Half}$ to be strictly larger than $D_0 + 1$ so that even after replacing the last $D_0$ unit-capacity edges of the route from $B_1$ to $d$, part of the $B_1$-to-$d$ path still has some regular unit-capacity edges (the thick lines in Fig. 11(a)). In our construction, we implicitly assume that node $B_1$ is the immediate downstream neighbor of the node where paths $su_0w_0d$ and $su_1w_1d$ diverge. The constant $2$ in (54) takes into account the edge from the diverging node to $B_1$ and the edge following $B_1$. Note that in (53), we only require that the length from the diverging node through $w_1$ to the entrance node of the $\alpha_1G^o$-subnetwork to be at least $1$. The reason is that the detour using node $w_1$ does not have the intermediate node $B_1$ and thus the feasibility condition (53), which aims to preserve the topology of the original network, can be slightly looser for the upper detour in the right triangle. More specifically, to preserve the topology of Fig. 11(a) (and also Fig. 11(b)) the entrance node of subnetwork $\alpha_1G^o$ can coincide with node $w_1$ but the entrance node of subnetwork $\alpha_2G^o$ cannot coincide with node $B_1$.

The intuition behind Conditions 2 and 3 will be explained shortly after.

In Appendix F, we will prove that for any given T/D spread $(\text{gain}^o, \Delta^o_L, \Delta^o_H)_D^o$ value, we can always find four strictly positive integer parameter values $\alpha_1$, $\alpha_0$, $\delta_1$, and $\text{Right.Half}$ satisfying (53) to (57). For example, if $(\text{gain}^o, \Delta^o_L, \Delta^o_H)_D^o = (\frac{1}{4}, 1, 1)$, then we can choose $\alpha_1 = 1$, $\alpha_0 = 3$, $\delta_1 = 2$, and $\text{Right.Half} = 20$, which satisfy (53) to (57).

After fixing the $\alpha_1$, $\alpha_0$, $\delta_1$, and $\text{Right.Half}$ values, the routing-equivalent network (Fig. 11(a)) is uniquely determined. We now convert the routing-equivalent network to a compound network by replacing each special path of length $D_1 = \alpha_1D^o$ by an $\alpha_1$-elongated version of $G^o$ for all $i \in \{0, 1\}$. See Fig. 11(b) for illustration. We now have the following lemma.

**Lemma 5:** The compound network Fig. 11(b) has delay-constrained NC capacity 2 packets per time slot, and its delay-constrained routing capacity is equal to that of Fig. 11(a).

Using this lemma, we can compute the delay-constrained routing capacity of Fig. 11(b) by solving the delay-constrained routing capacity of Fig. 11(a). By (52) with the value of $m = 2$, one can prove that, assuming $\text{gain}^o \leq 1.5$, the routing capacity of Fig. 11(a) is $\frac{0.5}{\text{gain}^o} + 1$. The new NC gain is $\text{gain} = \frac{2\text{gain}^o}{0.5 + \text{gain}^o}$. Our construction without the genie has the same NC gain as the genie-aided construction in Appendix D-B.

**Proof of Lemma 5:** The first half of Lemma 5 can be proven by the same NC scheme as discussed in Appendix D-B. Namely, we perform the NC scheme previously described in Section IV-B and then when traversing over the $\alpha_0G^o$ and $\alpha_1G^o$ subnetworks, instead of forwarding the packets, we perform the optimal NC solution associated to the constituent $G^o$.

For the second half of Lemma 5, we will prove that Lemma 3 holds for our new construction as well. To that end, we notice that as long as Observations 1 to 3 in the proof of Lemma 3 hold, then Lemma 3 holds. One can easily see that Observation 1 holds naturally. In the following we prove that Observations 2 and 3 hold for Figs. 11(b) and 11(a) as well.

To prove Observation 2, we notice that the only deadline-violating path in the routing-equivalent network Fig. 11(a) is the path corresponding to the $su_1w_1d$. Therefore, we need to prove that in the compound network Fig. 11(b), any path of the form $su_1w_1(\alpha_1G^o)d$ must have length $> D$. By the definition of $\Delta^o_L$ in (43) and $D_1 = \alpha_1D^o$, any path connecting the input/output nodes of $\alpha_1G^o$ must have length no less than $D_1 - \alpha_1\Delta^o_L$ hops. Therefore, any path of the form $su_1w_1(\alpha_1G^o)d$ in Fig. 11(b) must have at least $(2 + \delta_1 + 1 + \text{Right.Half} + \delta_1 - D_1) + (D_1 - \alpha_1\Delta^o_L)$ hops where $(2 + \delta_1 + 1 + \text{Right.Half} + \delta_1 - D_1)$ is the number of hops from $s$ to the input node of $\alpha_1G^o$ using the path $su_1w_1(\alpha_1G^o)d$. Since (55) in Condition 2 is true, all those paths have length $> D$. Observation 2 is thus proven.

We now prove Observation 3. The first half of Observation 3 is straightforward since the two subnetworks $\alpha_0G^o$ and $\alpha_1G^o$ form a cut. To prove the second half, we first consider all the deadline-respecting paths (length $\leq D$) in the compound network (Fig. 11(b)) that use the $\alpha_1G^o$ network. Those paths must be of the form $su_0w_0d(\alpha_1G^o)d$ according to Observation 2. Since the partial path from $s$ to the input node of $\alpha_1G^o$ via $su_0w_0d$ has $(2 + 1 + (\text{Right.Half} + \delta_1) - D_1)$ number of hops, it implies that the sub-path within $\alpha_1G^o$ must have length $\leq D - (2 + 1 + \text{Right.Half} + \delta_1 - D_1)$. Since our construction satisfies (56), Observation 3 holds for all deadline-respecting paths in the compound network that use $\alpha_1G^o$.

Let us now consider all the deadline-respecting paths (length $\leq D$) in the compound network (Fig. 11(b)) that use the $\alpha_0G^o$ network. We first quantify the shortest possible distance from $s$ to the input terminal of $\alpha_0G^o$ as follows. Since the $(A_1, B_1)$ pipe has the same length as the distance from $A_1$ to $B_1$ in the primary component, we only need to consider the shortest path from $s$ to the input terminal of $\alpha_0G^o$ in the primary component without using the $(A_1, B_1)$ pipe. One can easily see that such a shortest path must use the $su_0w_0d$ path. As a result, the shortest possible distance from $s$ to the input terminal of $\alpha_0G^o$ is $(2 + 1 + \text{Right.Half} - D_0)$ hops. This implies that for any deadline-respecting path in the compound network, the corresponding sub-path within $\alpha_0G^o$ must have length $\leq D - (2 + 1 + \text{Right.Half} - D_0)$. Since our construction satisfies (57), Observation 3 holds for all deadline-respecting paths in the compound network that use $\alpha_0G^o$.

Since Observations 1 to 3 hold for our new construction, the second half of Lemma 5 is proven.

**B. The generalized iterative construction without the help of a genie**

Appendix E-A is a parallel version of Appendix D-B, where the latter is based on the help of a genie and the former is not. Similarly, Appendix E-B generalizes Appendix D-C in the sense that even without the help of a genie, we can still
then the length of the path along the upper edges of the biggest right triangle is \( \text{Right.Half} + \delta_{m-1} \). If \( i = m - 2 \), then the detour path starts from the head of \( e^* \), continues along the straight path to the first branching point, then diverts to \( w_{m-2} \), and finally arrives at \( d \). Our construction requires that such a detour path has length \( \text{Right.Half} + \delta_{m-2} \).

Because our construction has to satisfy \( \text{length}(su_iw_{m-1-i}d) \) being the same for all \( i \in [0, m - 1] \), see \((36)\), we set the length of the left detour using \( u_i \) to be \( 2(m - 1) + \delta_{m-1} - \delta_{m-1-i} \), where we define \( \delta_0 = 0 \) and we count the detour distance as the length from the source \( s \) to the tail of the bottleneck edge \( e^* \) using node \( u_i \). For example, if \( i = m - 3 \), then the detour path starts from \( s \), uses the path corresponding to \( u_{m-3} \), merges with the straight line \( su_3w_0d \), and finally arrives at the tail of \( e^* \). Our construction requires that such a detour path has length \( 2(m - 1) + \delta_{m-1} - \delta_{m-1-i} = 2(m - 1) + \delta_{m-1} - \delta_{2} \).

The delay requirement value is set to be
\[
D = \text{length}(su_iw_{m-1-i}d) = (2(m - 1) + \delta_{m-1} - \delta_{m-1-i}) + 1 + (\text{Right.Half} + \delta_{m-1-i}) = 2(m - 1) + 1 + (\text{Right.Half} + \delta_{m-1}).
\]
See the high-level network description in \((38)\) and see Fig. 12(a) for illustration.

Once the basic topology of the network is fixed, we also require that for all \( i \in [1, m - 1] \), the last \( D_i \) unit-capacity edges of the route from \( w_i \) to \( d \) are replaced by a path with length \( D_i \) and capacity \( \frac{1}{\text{gain}_1} \). Similarly, the last \( D_0 \) unit-capacity edges of the straight route (directly from head \( e^* \) to \( d \) without using any detour) is replaced by a path with length \( D_0 \) and rate \( \frac{1}{\text{gain}_0} \). See Fig. 12(a) for illustration.

One can clearly see that the above construction is uniquely determined once the \( 2m \) integer values: \( \{\alpha_i : i \in [0, m - 1]\} \), \( \{\delta_j : j \in [1, m - 1]\} \), and Right.Half are fixed. We now describe how to choose these parameter values. In particular, we require them to satisfy the following conditions, for which the integer constant \( D \) is defined in \((58)\) and \( D_i \) defined in \((59)\), \( \forall i \in [0, m - 1] \):

- **Feasibility Condition:**
  \[
  \delta_{m-1} > \delta_{m-2} > \cdots > \delta_1 > \delta_0 = 0
  \]
  (69)
  Right.Half + \( \delta_{m-1} \geq D_{m-1} + 1 \);
  (60)
  Right.Half + \( \delta_i \geq D_i + (m - i) + 1, \forall i \in [1, m - 2]; \)
  (61)
  Right.Half \geq D_0 + (m - 1) + 1
  (62)

- **Condition 2:** For all \( j \in [1, m - 1] \),
  \[
  (2(m - 1) + \delta_{m-1} - \delta_{j-1}) + 1 + (\text{Right.Half} + \delta_j - D_j)
  + (D_j - \alpha_j \Delta^2_H) > D
  \]
  \( \iff \delta_{j-1} < \delta_j - \alpha_j \Delta^2_H \)
  (63)

where \((65)\) follows from \((63)\) by replacing \( D \) and \( D_i \) with their respective definitions.

- **Condition 3:** For all \( i \in [0, m - 1] \),
  \[
  D - (2(m - 1) + 1 + (\text{Right.Half} + \delta_i - D_i)) < D_i + \alpha_i \Delta^2_H
  \]
  \( \iff \delta_{m-1} - \alpha_i \Delta^2_H < \delta_1 \)
  (65)
where (66) follows from (65) by replacing $D$ and $D_i$ with their respective definitions.

The feasibility conditions are to ensure that the path length values in Fig. 12(a) are consistent. Specifically, (59) ensures that the larger the triangle in the illustration, the longer the detour length. To explain (60), we consider the detour using $w_{m-1}$, which is of length $\text{Right.Half} + \delta_{m-1}$. The feasibility condition (60) ensures after replacing the last $D_{m-1}$ unit-capacity edges of the detour by a special path with length $D_{m-1}$ and capacity $1/gain_i$, we still have part of the $w_{m-1}$ detour being the regular unit-capacity edges (the thick lines in Fig. 12(a)). Similarly, for all $i = m-2, m-3, \ldots, 1$, the detour using $w_i$ has length $\text{Right.Half} + \delta_i$. We then note that the path from the head of $e^*$ to node $B_{m-1-i}$ has $(m-i)$ hops. The feasibility condition (61) ensures that after replacing the last $D_i$ unit-capacity edges of the route from $w_i$ to $d$ by a special path with length $D_i$ and capacity $1/gain_i$, we still have at least one edge after node $B_{m-1-i}$ being the regular unit-capacity edges (the thick lines in Fig. 12(a)).

When not detouring at all in the right triangles, the direct path from head($e^*$) to $d$ has length $\text{Right.Half}$. We also note that the path from the head of $e^*$ to node $B_{m-1}$ has $(m-1)$ hops. The feasibility condition (62) then ensures that after replacing the last $D_0$ unit-capacity edges of the route from $B_{m-1}$ to $d$ by a special path with length $D_0$ and capacity $1/gain$, the $B_{m-1}$-to-$d$ path still has at least one regular unit-capacity edge (the thick lines in Fig. 12(a)). Conditions 2 and 3 will be explained shortly after.

In Appendix F, we will prove that for any given T/D spread $(gain^n, \Delta_L^n, \Delta_H^n)$, value, we can always find $2m$ parameter values $\{\alpha_i > 0 : i = 0, 1, \ldots, m-1\}$, $\{\delta_j > 0 : j = 1, \ldots, m-1\}$, and $\text{Right.Half} > 0$ satisfying (59) to (66). Therefore, the construction of the routing-equivalent network (Fig. 12(a)) in Step 2 is always feasible.

**Step 3:** After constructing the routing-equivalent network (Fig. 12(a)) described in Step 2, we convert it to a compound network (Fig. 12(b)) by replacing each special path of length $D_i = \alpha_i D^0$ by an $\alpha_i$-elongated version of $G^0$ for all $i \in [0, m-1]$ and keep the same delay requirement $D$ value defined in (58). The description of the new compound network instance $(G, D)$ is now complete.

The two networks constructed in Steps 1 to 3 satisfy the following lemma.

**Lemma 6:** The compound network Fig. 12(b) has delay-constrained NC capacity $m$ packets per time slot, and its delay-constrained routing capacity is equal to that of Fig. 12(a).

Using this lemma, we can compute the delay-constrained routing capacity of Fig. 12(b) by solving the delay-constrained routing capacity of Fig. 12(a). By (52), one can prove that, assuming $gain^n \leq 2 - 2^{-(m-1)}$, the routing capacity is $m - 2 + 2^{-(m-1)} + 1$. The new NC gain is $gain = \frac{m - 2 + 2^{-(m-1)} + 1}{gain^n}$. Our construction without the genie has the same NC gain as the genie-aided construction in Appendix D-C. By iteratively applying Steps 2 to 3 in this section, we can design a network with NC gain $> 2 - \epsilon$ for any given $\epsilon > 0$. Our construction is complete.

**Proof of Lemma 6:** The first half of Lemma 6 can be proven by the same NC scheme as discussed in Appendix D-C. Namely, we perform the NC scheme previously described in Appendix C-B and then when traversing over the $\alpha_i G^0$ subnetworks, instead of forwarding the packets, we perform the optimal NC solution associated to the constituent $G^0$.

For the second half of Lemma 6, we will prove that Lemma 3 holds for our new construction as well. To that end, we notice that as long as Observations 1 to 3 in the proof of Lemma 3 hold, then Lemma 3 holds. One can easily see that Observation 1 holds naturally. In the following we prove that for any arbitrarily given $m \geq 2$, Observations 2 and 3 hold in our new genie-free construction. See Figs. 12(b) and 12(a) for illustration.

To prove Observation 2, we notice that the deadline-violating path in the routing-equivalent network Fig. 12(a) must be of the form $su_j w_j d$ with $i + j \geq m$. As a result, we need to prove that any path of the form $su_j w_j \alpha(G) d$, $i + j \geq m$, in the compound network must have length $> D$. Since the larger the triangle the longer the detour distance, we only need to prove that any path of the form $su_{m-j} w_j \alpha(G) d$, $j \in [1, m-1]$ in the compound network must have length $> D$.

By the definition of $\Delta_L^n$ in (43) and $D_i = \alpha_i D^0$, any path connecting the input/output nodes of $\alpha_i G^0$ must have length no less than $D_i - \delta_j \Delta_L^n$ hops. Therefore, any path of the form $su_{m-j} w_j \alpha(G) d$ in Fig. 12(b) must have at least $(2(m-1) - \delta_{m-1} - \delta_{m-1}) + 1$ hops. Therefore, the new NC gain is $\frac{2(m-1) - \delta_{m-1} - \delta_{m-1}}{gain^0}$, in the compound network must have length $> D$.

We now prove Observation 3. The first half of Observation 3 is straightforward since the subnetworks $\alpha_j G^0$, $j \in [0, m-1]$ form a cut. To prove the second half, we first consider all the deadline-respecting paths (length $\leq D$) in the compound network (Fig. 12(b)) that use the $\alpha_j G^0$ network for some $i \in [0, m-2]$. The case in which $i = m-1$ is a degenerate case and follows similarly. We first quantify the shortest possible distance from $s$ to the input terminal of $\alpha_i G^0$ as follows. Since the $(A_{m-1-i}, B_{m-1-i})$ pipe has the same length as the distance from $A_{m-1-i}$ to $B_{m-1-i}$ in the primary component, we only need to consider the shortest path from $s$ to the input terminal of $\alpha_i G^0$ in the primary component without using the $(A_{m-1-i}, B_{m-1-i})$ pipe. One can easily see that such a shortest path must use the $su_{m-i} d$ path. As a result, the shortest possible distance from $s$ to the input terminal of $\alpha_i G^0$ is $(2(m-1) + \delta_{m-1} - \delta_{m-1}) + 1$ hops. This implies that for any deadline-respecting path in the compound network, the corresponding sub-path within $\alpha_i G^0$ must have length $\leq D - (2(m-1) + 1 + \text{Right.Half} + \delta_{m-1} - D_j)$. Since our construction satisfies (65), Observation 3 holds for all deadline-respecting paths in the compound network that use $\alpha_j G^0$.

Since Observations 1 to 3 hold for our new construction, the second half of Lemma 6 is proven.
APPENDIX F
THE FEASIBILITY OF THE GENERAL COMPOUND NETWORK CONSTRUCTION IN APPENDIX E-B

In this section, we will prove that we can always find 2m integer values \( \{\alpha_i > 0 : i \in [0, m-1]\} \) and \( \{\delta_j > 0 : j \in [1, m-1]\} \), and Right.Half > 0 satisfying (59) to (66). We notice that (64) automatically implies (59) and we will thus focus only on (60) to (66) subsequently.

To that end, we first notice that Right.Half appears only on the left-hand sides of (60) to (62), not in (64) nor in (66). Therefore, whenever we have finished choosing the \( \{\alpha_i\} \) and \( \{\delta_j\} \) values, we can always choose a sufficiently large Right.Half to satisfy (60) to (62).

Before describing how to choose 2m − 1 integer values \( \{\alpha_i > 0 : i \in [0, m-1]\} \) and \( \{\delta_j > 0 : j \in [1, m-1]\} \), we relax the problem a bit and focus on finding 2m integer values \( \{\alpha_i > 0 : i \in [0, m-1]\} \) and \( \{\delta_j : j \in [0, m-1]\} \) that satisfy (64) and (66). Namely, instead of focusing on finding \( \delta_1 \) to \( \delta_{m-1} \) with the value of the dummy variable \( \delta_0 \) hardwired to 0, we now are allowed to choose the \( \delta_0 \) value. Furthermore, previously all \( \delta_j \) values have to be strictly positive. In our relaxed problem, we allow negative \( \delta_j \). We will first prove that with such a relaxation, finding those \( \{\alpha_i\} \) and \( \{\delta_j\} \) values is always possible.

We prove the above claim by explicit construction. We first set \( \delta_{m-1} = 0 \) and \( \alpha_{m-1} = 1 \). Then for \( i = m-2 \) back to 0, we set the \( \delta_i \) and \( \alpha_i \) in the following sequential way. For any given \( i \), we choose \( \delta_i \) as an integer satisfying (64). This is always possible since we have already fixed our choices of \( \delta_{i+1} \) and \( \alpha_{i+1} \) in the previous round and we allow \( \delta_i \) to take a negative value. After deciding the \( \delta_i \) value, we choose the \( \alpha_i \) value as a strictly positive integer satisfying (66). Again, this is always possible since \( \delta_i \) has been decided already and by our definition of \( \Delta_{ji} \) in (44) we always have \( \Delta_{ji} \geq 1 \). The above procedure is repeated for all \( i = m-2 \) back to 0 and we have thus found the desired \( \{\alpha_i > 0 : i \in [0, m-1]\} \) and \( \{\delta_j : j \in [0, m-1]\} \) satisfying (64) and (66) simultaneously.

Now we describe how to incorporate the positivity constraint on \( \delta_j \) and the dummy constant constraint \( \delta_0 = 0 \). To that end, we first use \( \delta_{j, \text{old}} \) to denote the \( \delta_j \) value we found in the previous construction. Define \( x \triangleq \delta_{0, \text{old}} \). Then we keep the same \( \{\alpha_i > 0 : i \in [0, m-1]\} \) values of our previous construction but choose the new \( \delta_{j, \text{new}} \) by \( \delta_{j, \text{new}} = \delta_{j, \text{old}} - x \) for all \( j \in [0, m-1] \).

It is clear that \( \delta_{0, \text{new}} = 0 \) satisfies the dummy constant constraint. Also, the new \( \delta_{j, \text{new}} \) and the previously constructed \( \alpha_i > 0 : i \in [0, m-1]\) must satisfy both (64) and (66) since we now shift all the \( \delta_j \) values by the same \( x \) amount. Finally, since the new \( \delta_{j, \text{new}} \) satisfy (64), they are strictly increasing with respect to \( j \). Therefore, \( \delta_{j, \text{new}} > \delta_{j, \text{new}} = 0 \) for all \( j \in [1, m-1] \). The positivity condition on \( \delta_j \) also holds.

The above explicit construction shows that we can always find the integer values \( \{\alpha_i > 0 : i \in [0, m-1]\} \), \( \{\delta_j > 0 : j \in [1, m-1]\} \), and Right.Half > 0 satisfying (60) to (66). The proof is complete.

APPENDIX G
PROOFS OF PROPOSITION 4 AND COROLLARY 1

Proof of Proposition 4: Consider any delay-constrained rate \( R \) that is feasible. By Proposition 1, for any arbitrary \( T \) value, the time expanded graph \( G^{[T+D]} \) can sustain \( T \) simultaneous unicast flows from \( [s,t] \) to \( [d,t+D] \) for all \( t \in [1, T] \) with individual rate \( R \).

Consider any arbitrary \( T \) satisfying \( T > L \). We define

\[
E_{h,T} \triangleq \{(u,v) : (u,v) \in E, \forall t \in [1, T+D] \} \text{ s.t. } h(e, \text{ mod } (t, L)) = 1
\]

in the time expanded graph \( G^{[T+D]} \), where \( h(\cdot, \cdot) \) is a function satisfying the statements in Proposition 4. The difference between \( E_{h,T} \) and the previously defined \( E_h \) in (23) is that \( E_{h,T} \) is defined for arbitrarily large \( T \) while \( E_h \) is defined for a given \( L \).

We now prove that \( E_{h,T} \) in (67) is an edge-cut in \( G^{[T+D]} \) that separates \( [s,t] \) from \( [d,t+D] \) for all \( t \in [1,T] \). Suppose not. Then there exists \( t_1 \leq T \) and \( t_2 \leq t_1 + D \) such that there exists a path from \( [s,t_1] \) to \( [d,t_2] \) in \( G^{[T+D]} \) without using any edge in \( E_{h,T} \). Note that by the “causality” used in the construction of \( G^{[T+D]} \), this implicitly implies \( t_1 < t_2 \).

We now argue that if such a path exists, there exists a path from \( [s,t_1] \) to \( [d,t_2] \) in \( G^{[L+D]} \) without using \( E_h \) where \( t_1 \triangleq \text{mod}(t_1 - 1, L) + 1 \) and \( t_2 \triangleq t_1 + t_2 - t_1 \). The reason is that whatever the path form \( [s,t_1] \) to \( [d,t_2] \) in \( G^{[T+D]} \) is, we can transcribe it to a path from \( [s,t_1] \) to \( [d,t_2] \) in \( G^{[T+D]} \) by shifting the time indices of the intermediate nodes by \( t_1 - t_1 \). The time-shifted new path does not use any edge in \( E_{h,T} \) since the shift amount \( t_1 - t_1 \) is a multiple of \( L \) and \( E_{h,T} \) includes edges “periodically with period \( L \)” (see (67)). Furthermore, such a path is not only in \( G^{[L+D]} \) but also in \( G^{[T+D]} \) since by our construction \( t_1 \leq L \) and \( t_2 \leq L + D \). By noting that \( E_{h,T} \) in (67) is a superset of \( E_h \) in (23), the time-shifted new path does not use \( E_h \), either.

However, the existence of such a path contradicts the statement in Proposition 4 that \( E_{h,T} \) forms an edge-cut separating \( [s,t] \) from \( [d,t+D] \) for all \( t \in [1,L] \) since one can now traverse from \( [s,t_1] \) to \( [d,t_2] \) and then traverse through \( [d,t_2] \) to \( [d,t_2+1] \) to \( \cdots \) to \( [d,t_1 + D] \) without using \( E_h \). This contradiction proves that \( E_{h,T} \) in (67) is an edge-cut in \( G^{[T+D]} \) that separates \( [s,t] \) from \( [d,t+D] \) for all \( t \in [1,T] \).

Since \( E_{h,T} \) is an edge cut separating \( [s,t] \) from \( [d,t+D] \) for all \( t \in [1,T] \) by the generalized network-sharing bound in (21), we have

\[
T \cdot R \leq \sum_{e \in E_{h,T}} c_e \leq \left[T + D - 1 \right] \frac{L}{R} \sum_{t=0}^{T} \left( \sum_{e \in E(t,e)} c_e \right)
\]
where (68) follows from that the sum rate of the $T$ coexisting flows is no larger than the generalized cut set value [21]; and (69) follows from the definition of $E_{h,T}$ in (67) and from over-counting one extra time period of $L$ time slots.

Ineq. (69) implies

$$R \leq \frac{L}{T} \left[ T + D - 1 \right] \sum_{e} \left( \frac{\sum_{t=0}^{T-1} h(e,t)}{L} \right) c_e, \forall T.$$ 

By letting $T \to \infty$, we have proven (24).

Proof of Corollary 1: Specifically, consider any IP solution $\{y^*_e\}$ that leads to an upper bound in Proposition 3. By setting $L = 1$ and the binary mapping to be $h(e,0) = y^*_e$ for all $e$, one can verify that the $E_h$ in (23) is an edge-cut in $G^{1+[D]}$ that separates $[s,1]$ from $[d,1+D]$ since $\{y^*_e\}$ corresponds to a cut in $G$ severing all paths of length $\leq D$. One can easily verify that the resulting upper bound in (24) for this particular choice of $L = 1$ and $h(e,0) = y^*_e$ is identical to (17). As a result, any upper bound in Proposition 3 can be used to construct an upper bound in Proposition 4 with the same value. The equivalence is thus proven.

**APPENDIX H**

**The Generalized Cut Set Verification**

This section considers exclusively Fig. 3(b). To show that the $E_h$ in (23) generated by $L = 2$ and the $h(e,t)$ in (25)–(26) is an edge cut separating $[s,t]$ from $\{[d,t+D] : \forall \tau \in [1,t]\}$ for all $t \in [1,2]$, we notice that there are only 4 types of paths that connect $[s,t]$ to $[d,t+D]$, where $D = 6$. They are

Type 1: The path $[s,t] \to [v_1, t+1] \to [v_2, t+2] \to [v_3, t+3] \to [v_7, t+4] \to [v_8, t+5] \to [d, t+6]$. If $t = 2$, then the edge $[s,t] \to [v_1, t+1]$ will be included in $E_h$ by (25). If $t = 1$, then the edge $[v_2, t+2] \to [v_3, t+3]$ will be included in $E_h$ by (26).

Type 2: The path $[s,t] \to [v_5, t+1] \to [v_6, t+2] \to [v_2, t+3] \to [v_3, t+4] \to [v_4, t+5] \to [d, t+6]$. If $t = 2$, then the edge $[v_2, t+3] \to [v_3, t+4]$ will be included in $E_h$ by (26). If $t = 1$, then the edge $[v_4, t+5] \to [d, t+6]$ will be included in $E_h$ by (25).

Type 3: The path $[s,t] \to [v_9, t+1] \to [v_{10}, t+2] \to [v_{11}, t+3] \to [v_{10}, t+4] \to [v_4, t+5] \to [d, t+6]$. If $t = 2$, then the edge $[s,t] \to [v_9, t+1]$ will be included in $E_h$ by (25). If $t = 1$, then the edge $[v_4, t+5] \to [d, t+6]$ will be included in $E_h$ by (25).

Type 4: Type 4 corresponds to the paths of the form $sv_1v_2v_3v_4d$. Since the path $sv_1v_2v_3v_4d$ in the original graph $G$ has length 5, it means that if we go directly from $[s,t]$ to $[d,t+6]$ in the time expanded graph $G^{[L+D]}$, the packet will arrive in 5 time slots rather than 6. Therefore, we can “wait” and stay idle in one of the six nodes $\{s, v_1, v_2, v_3, v_4, d\}$. For example, if we stay in $v_1$, then the corresponding path in $G^{[L+D]}$ becomes $[s,t] \to [v_1, t+1] \to [v_1, t+2] \to [v_2, t+3] \to [v_3, t+4] \to [v_4, t+5] \to [d, t+6]$. If we stay in $d$, then the corresponding path in $G^{[L+D]}$ becomes $[s,t] \to [v_1, t+1] \to [v_2, t+2] \to [v_3, t+3] \to [v_4, t+4] \to [d, t+5] \to [d, t+6]$. There are 6 paths of type-4, each corresponding to staying in one of the six nodes $\{s, v_1, v_2, v_3, v_4, d\}$, respectively.

We now consider two cases. Case 1: If we stay idle in one of $\{v_3, v_4, d\}$, then the first 3 edges of the paths must be $[s,t] \to [v_1, t+1] \to [v_2, t+2] \to [v_3, t+3] \to [v_4, t+4] \to [d, t+5] \to [d, t+6]$. If $t = 2$, then the edge $[s,t] \to [v_1, t+1]$ will be included in $E_h$ by (25). If $t = 1$, then the edge $[v_2, t+2] \to [v_3, t+3]$ will be included in $E_h$ by (26).

Case 2: If we stay idle in one of $\{s, v_1, v_2\}$, then the last 3 edges of the paths must be $[v_2, t+3] \to [v_3, t+4] \to [v_4, t+5] \to [d, t+6]$. If $t = 2$, then the edge $[v_2, t+3] \to [v_3, t+4]$ will be included in $E_h$ by (26). If $t = 1$, then the edge $[v_4, t+5] \to [d, t+6]$ will be included in $E_h$ by (25).

The above analysis shows that $E_h$ separates $[s,t]$ from $[d, t+6]$ for all $t \in [1,2]$. Our proof is thus complete.

**APPENDIX I**

**A Proof of Proposition 5**

For any $m \geq 2$, we construct a network instance as follows. See Fig. 13 for illustration. The network contains three major components.

Component 1: A direct path of length $m$ connecting $s$ to $d$. We denote the intermediate nodes by $v_i, \forall i \in [1, m-1]$.

Component 2: A set of $m-1$ paths connecting $s$ and $v_i$ for $i \in [1, m-1]$. Each path is of length $2i$. Component 2 is illustrated by the paths in the upper half of Fig. 13. For future references, the path connecting $s$ and $v_i$ is denoted by $U_i$. Namely, the $i$-th Upper path.

Component 3: A set of $m-1$ paths connecting $v_i$ and $d$ for $i \in [1, m-1]$. Each path is of length $2(m-i)$. Component 3 is illustrated by the paths in the lower half of Fig. 13. For future references, the path connecting $v_i$ and $d$ is denoted by $L_i$. Namely, the $i$-th Lower path.

All edges/paths are of unit capacity, i.e., $c_e = 1, \forall e \in E$. The delay requirement is set to $D = 2m-1$.

We first show that $P_{route} = m$ for the above network. The reason is that the following $m$ paths

$$sv_1L_1d$$
$$sU_i v_i + 1 d, \forall i \in [1, m-2]$$
$$sU_{m-1} v_{m-1} d$$
all have length \(2m-1\) and are edge-disjoint. As a result, we have \(R_{\text{route}}^* = m\) since the min-cut value from \(s\) to \(d\) is \(m\).

We now prove that \(R_{\text{RLNC}}^* = 1\). To that end, we first consider the transfer function of the network, which takes into account the local coding coefficients in (29), but omits the impact of the preceding operations in (28). More specifically, suppose at time \(t\) source \(s\) sends a coded symbol \(W_i(t)\) directly to \(v_i\) and sends coded symbols \(W_{i+1}(t)\) through the path \(U_i\) for \(i \in [1, m-1]\). And also suppose in the end of time \(t\) destination \(d\) receives \(Y_m(t)\) directly from \(v_{m-1}\) and receives \(Y_i(t)\) through the path \(L_i\) for \(i \in [1, m-1]\). Then the transfer function between the input vector \(\tilde{W}(t) = (W_1(t), \ldots, W_m(t))^T\) and the output vector \(\tilde{Y}(t) = (Y_1(t), \ldots, Y_m(t))^T\) can be written as

\[
\tilde{Y}(t) = \sum_{\tau=1}^{t} F_{i,\tau} \tilde{W}(\tau)
\]

where \(F_{i,\tau}\) is the transfer matrix from \(\tilde{W}(\tau)\) to \(\tilde{Y}(t)\), which is a function of the local coding coefficients of the network nodes. Since we assume that the local coding coefficients are fixed and do not change over time, \(F_{i,\tau}\) is a function of the time difference \(\Delta = t - \tau\). We can thus define \(F_{\Delta} = F_{\tau+\Delta,\tau}\).

Using the above notation of the transfer matrix \(F_{\Delta}\), [14] proves the following results regarding the delay-constrained throughput.

**Proposition 8 ([14, Theorem 2]):** For any given set of local coding coefficients, we consider its transfer matrix \(F_{\Delta}\). For any given integer \(x \in [0, \infty)\), define the following matrix

\[
F_{\Delta}^x \triangleq \begin{bmatrix}
F_0 & 0 & \cdots & 0 \\
F_1 & F_0 & 0 & \cdots \\
F_2 & F_1 & F_0 & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
F_x & F_{x-1} & F_{x-2} & \cdots & F_0
\end{bmatrix}.
\]  

(70)

Then the largest supportable rate under the delay constraint \(D\), assuming both the preceding vectors (how we generate \(\tilde{W}(t)\)) and the decoding operations at \(d\) are chosen in the optimal way, is characterized by

\[R^* = \text{Rank}(F_{D-1}) - \text{Rank}(F_{D-2}).\]  

(71)

We will use Proposition 8 to prove that \(R_{\text{RLNC}}^* = 1\) in Fig. 13. To that end, we first assume that all the interior nodes of paths \(U_i\) and \(L_i\) only perform pure relaying. Such an assumption is without loss of generality since with a sufficiently large finite field \(GF(q)\), the RLNC performed by any node \(u\) with \(|\ln(\mu)| = 1\) is equivalent to “relaying” with close-to-one probability. And we denote the local coding coefficients of node \(v_i\), \(i \in [1, m-1]\), by

\[
\beta_{U,H}^{[i]}, \beta_{U,H}^{[i]}, \beta_{U,L}^{[i]}, \text{ and } \beta_{U,L}^{[i]}.
\]  

(72)

Namely, \(\beta_{U,H}^{[i]}\) is the coding coefficient used by \(v_i\) when going from the horizontal input edge to the horizontal output edge of \(v_i\); \(\beta_{U,H}^{[i]}\) is the coding coefficient used by \(v_i\) when going from the Upper path \(U_i\) to the horizontal output edge; \(\beta_{U,L}^{[i]}\) is the coding coefficient used by \(v_i\) when going from the Horizontal input edge to the Lower path \(L_i\); and \(\beta_{U,L}^{[i]}\) is the coding coefficient used by \(v_i\) when going from the Upper path \(U_i\) to the Lower path \(L_i\).

Using the local coding coefficients in (72), we will characterize the expressions of \(F_{\Delta}\) in Fig. 13. We first notice that since the shortest path from \(s\) to \(d\) is of length \(m\), we have \(F_{\Delta} = 0\) if \(\Delta \leq m - 2\). We now characterize \(F_{\Delta}\) when \(\Delta \in [m-1, D-1] = [m-1, 2m-2]\). Specifically, we can write down the transfer matrix \(F_{\Delta}\) by

\[
F_{\Delta} \triangleq \begin{cases}
\beta_{U,H}^{[i]j} \left( \prod_{l=1}^{[j]} \beta_{H,H}^{[i]} \right) \beta_{H,L}^{[i]} & \text{if } i - j = 2m - 2 - \Delta \\
0 & \text{otherwise}
\end{cases}
\]  

(73)

Here we use the convention that \(\beta_{U,H}^{[0]} = 1 = \beta_{H,L}^{[m-1]}\) since the coding coefficients in (72) are defined only for \(i \in [1, m-1]\).

Define

\[
\forall j \in [1, m-1], \quad \gamma^{[i]} \triangleq \frac{\beta_{U,H}^{[i]j}}{\beta_{U,H}^{[i]j+1} \beta_{H,H}^{[i]}}, \quad \text{and} \quad \Gamma \triangleq \begin{bmatrix}
\gamma^{[1]} & 0 & \cdots & 0 \\
0 & \gamma^{[2]} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \gamma^{[m-1]}
\end{bmatrix}.
\]

Since all the coding coefficients in (72) are chosen randomly, \(\gamma^{[i]}\) and \(\Gamma\) are well-defined with close-to-one probability when the underlying finite field \(GF(q)\) is sufficiently large.

By (73), one can easily verify that

\[
\begin{bmatrix}
F_{m-1} \\
F_m \\
F_{m+1} \\
\vdots \\
F_{2m-2}
\end{bmatrix} \begin{bmatrix}
0_{1 \times (m-1)} \\
I_{(m-1) \times (m-1)} \\
\vdots \\
F_{2m-3}
\end{bmatrix} = \begin{bmatrix}
0 \\
F_{m-1} \\
F_m \\
\vdots \\
F_{2m-3}
\end{bmatrix} \Gamma
\]

(74)

where \(0_{1 \times (m-1)}\) is a zero row vector of dimension \(m-1\) and \(I_{(m-1) \times (m-1)}\) is the \((m-1)\)-by-\((m-1)\) identity matrix.

If we remove the first column of \(F_{D-1}\) and denote the remaining \((mD) \times (mD-1)\) matrix as \(F_{D-1}'\), we will then have \(\text{Rank}(F_{D-1}') = \text{Rank}(F_{D-2}')\) since (i) \(F_{\Delta} = 0\) if \(\Delta \leq m - 2\); (ii) by (70) and (74) one can see every column of \(F_{D-1}'\) can be written as a linear combination of the columns in \(F_{D-2}'\) and (iii) we also have \(\text{Rank}(F_{D-2}') = \text{Rank}(F_{D-2})\). The fact that \(\text{Rank}(F_{D-1}) = \text{Rank}(F_{D-2})\) then implies

\[
\text{Rank}(F_{D-1}) - \text{Rank}(F_{D-2}) \leq 1
\]
whenever $\Gamma$ is well-defined, which is of close-to-one probability.

Finally, since the $(m,1)$-th entry of $F_{m-1}$ being
\[ \prod_{j=1}^{m-1} \beta_{H,H}^{(j)} \] is non-zero with close-to-one probability, we have $\text{Rank}(F_{d-1}) - \text{Rank}(F_{d-2}) = 1$ with close-to-one probability. By Proposition 8, the largest supported $R_{RLNC} = 1$ even with the optimal precoder and decoder designs.

APPENDIX J

Fig. 4(a) Has $R_{RLNC} = 2$

In this section we show that with high probability, RLNC is able to support an integer rate $R = 2$ in Fig. 4(a). The proof that RLNC is not able to support $R = 3$ is similar to our discussion in Section II-C and is thus omitted.

For the ease of exposition, we assume that the message symbols for each time $t$ are $X(t)$ and $Y(t)$. Since the symbols are precoded in (28), we assume
\[ M(t)_{u_1,v_2} = a_1 X(t) + b_1 Y(t) \]
\[ M(t)_{u_1,v_3} = a_2 X(t) + b_2 Y(t) \]
\[ M(t)_{u_2,v_3} = a_3 X(t) + b_3 Y(t) \]
where the coefficients $a_i$ and $b_i$ are randomly chosen. Assuming nodes \{v_1, v_4, v_5, v_6, v_7\} only perform pure relaying, we will have
\[ M(t)_{v_2,d} = \beta_{s,d}(a_1 X(t - 1) + b_1 Y(t - 1)) + \beta_{v_1,d}(a_2 X(t - 2) + b_2 Y(t - 2)) + \beta_{v_2,d}(a_3 X(t - 3) + b_3 Y(t - 3)) \]
where for any $u_1 \in \text{In}(v_2)$ and $u_2 \in \text{Out}(v_2)$, the scalar $\beta_{u_1,u_2}$ is shorthand for the local coding coefficient from edge $(u_1, v_2)$ to edge $(v_2, u_2)$ at node $v_2$. Similarly, we have
\[ M(t)_{v_3,d} = \beta_{v_1,v_3}(a_1 X(t - 2) + b_1 Y(t - 2)) + \beta_{v_2,v_3}(a_2 X(t - 3) + b_2 Y(t - 3)) + \beta_{v_3,v_3}(a_3 X(t - 4) + b_3 Y(t - 4)) \]
\[ M(t)_{v_4,d} = \beta_{v_1,v_4}(a_1 X(t - 3) + b_1 Y(t - 3)) + \beta_{v_2,v_4}(a_2 X(t - 4) + b_2 Y(t - 4)) + \beta_{v_3,v_4}(a_3 X(t - 5) + b_3 Y(t - 5)) \]
Recall that $d$ would like to decode $X(t - 3)$ and $Y(t - 3)$ by the end of time $t$ since $D = 4$. By delaying the $M(t)$ coded symbol in (75), destination $d$ can compute
\[ M(t)_{v_2,d} = \frac{\beta_{s,v_3}}{\beta_{s,d}} M(t-1)_{v_2,d} \]
\[ = \left( \beta_{v_1,v_3} - \frac{\beta_{s,v_3}}{\beta_{s,d}} \beta_{v_1,d} \right)(a_2 X(t - 3) + b_2 Y(t - 3)) \]
\[ + \left( \beta_{v_2,v_3} - \frac{\beta_{s,v_3}}{\beta_{s,d}} \beta_{v_2,d} \right)(a_3 X(t - 4) + b_2 Y(t - 4)) \]
(78)

We now notice that in (77) and (78) the terms corresponding to $(X(t-4), Y(t-4))$ and $(X(t-5), Y(t-5))$ have already been decoded in the past, and they can thus be removed. Destination $d$ can then decode $(X(t-3), Y(t-3))$, with close-to-one probability, from the remaining terms of (77) and (78), which are
\[ \beta_{s,v_3}(a_2 X(t - 3) + b_1 Y(t - 3)) \]
and
\[ \left( \beta_{v_1,v_3} - \frac{\beta_{s,v_3}}{\beta_{s,d}} \beta_{v_1,d} \right)(a_2 X(t - 3) + b_2 Y(t - 3)) \]
Rate $R = 2$ is thus supported by RLNC.

REFERENCES


