On Constructive Network Coding for Multiple Unicasts

Tracey C. Ho, Yu-Han Chang and Keesook J. Han

***Abstract*— We consider the problem of network coding across multiple unicasts. We develop, for wired and wireless networks, off-line and online back pressure algorithms for finding approximately throughput-optimal network codes within the class of network codes restricted to XOR coding between pairs of flows. Our online algorithm incorporates real-time control signaling with delays, and random exploration approaches for reducing computation.**

I. Introduction

In this paper we consider network coding across multiple unicasts, using the class of pairwise XOR codes introduced in [12], [13] for wired networks. While this class of codes is not optimal in all cases[1], it covers a substantial number of common known cases including "reverse carpooling" and two-flow "star coding" for wireless networks [6]. In this class of codes, network coding is limited to XOR coding between pairs of uncoded packets. Two uncoded packets of different sessions can be coded together to form a *joint poison* packet in order to share capacity on one or more hops. The joint poison packet is subsequently replicated to form two identical *individual poison* packets whose routes branch (diverge). These are met by corresponding *remedy* packets and decoded to form the original uncoded packets. Decoded packets can be subsequently re-encoded.

We consider the problem of constructing throughput-optimal network codes within this class on wired and wireless networks. In an off-line setting we develop, for a given network and communication demands, a combinatorial approximation algorithm that finds a solution for the problem with rates (r_c) if there exists a solution for the problem with slightly higher rates $((1+2\epsilon)r_c)$ for any $\epsilon > 0$. In a dynamic online setting, we incorporate real-time control signaling with delays, and approaches for reducing computation through random exploration of the optimization space.

Our approach is inspired by back pressure techniques originally introduced for the multiple unicasts problem without coding [1], [14], which maintain a queue for each session's packets at each node, and route based on queue gradients that form by the addition of packets to sources and their removal from sinks. Extension of this approach to incorporate coding is not straightforward due to some significant complicating features. Firstly, since the path taken by a packet affects its future coding possibilities, uncoded packets of the same session that have arrived at the same node via different paths may have to be treated as separate commodities (where a commodity is a class of packets that are treated interchangeably from the standpoint of scheduling, routing and coding decisions, without affecting the throughput of each session).[2] Secondly, a poison packet is removed by its corresponding remedy packet when they meet at any node in the network, unlike the no coding case where all packets of a session are removed independently at a fixed location. Consequently, the choice of how different commodities are defined has important bearing on the optimality and complexity of the algorithm.

This work was supported by the Caltech Lee Center for Networking and AFOSR Grant 5710001972.

T. Ho is with the California Institute of Technology.
Y. Chang is with the University of Southern California.
K. J. Han is with the Air Force Research Laboratory.

[1]Linear coding is not sufficient in general for multiple unicasts [4], [5].

A. Other related work

For brevity we do not list many other works on network coding and on back pressure techniques in networking, but mention here a few that are most closely related to this work. Back pressure has previously been applied to multicast network coding [8]. Opportunistic XOR coding is proposed in [9]. A more general approach for multiple unicast network coding based on state space realizations is given in [13]. Constructive XOR coding across pairs of unicasts is considered in [13], [15] using a linear optimization approach. Independent work by [7] also considers back pressure for the same problem; while their algorithm superficially resembles ours, it differs in a number of significant aspects. For instance, [7] defines commodities solely by their destination and multicast group (poison flows are multicast), i.e. at each node i, packets intended for a particular destination node d within a multicast group D are placed in a queue $Q_i^{(d,D)}$. We define commodities quite differently, as described in the following sections. In [7] a coding node remotely chooses the remedy origination locations as well as the decoding locations based on the queue lengths at these locations; in our algorithms the decoding locations are not predetermined by the coding node but are chosen locally.

II. Problem statement and definitions

K unicast sessions are transmitted over a network represented as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ of $N = |\mathcal{N}|$ nodes and $M = |\mathcal{L}|$ links. We refer to a unit of flow as a packet.

The off-line version of the problem specifies the demanded communication rate r_c for each session $c = 1, \ldots, K$, and a set of link rates (in the wireline case) or link rate constraints (in the wireless case). A solution specifies different types of packets and the average rates at which they are transmitted, coded, decoded, etc., at different nodes and links.

[2]A poison packet must nonetheless be labeled with the identifiers of its constituent packets, since remedy and poison packets must be correctly matched for actual decoding purposes.

In the online version of the problem, the instantaneous source arrival rates and link capacities/constraints may vary ergodically. For simplicity of exposition we assume that the channel and arrival processes are independent and identically distributed across time slots; a straightforward generalization to ergodic processes is possible using a similar approach as that in [11]. Here we give a dynamic policy that depends on the state of the network.

In both cases, we show optimality of our algorithms by comparison with an assumed, but unknown, pairwise XOR-coding solution. A solution is made up of *elementary flows* such that all packets in an elementary flow undergo the same routing and coding operations. Each elementary flow has a set of links comprising a single *primary path* from s_c to d_c and a remedy path associated with each node at which decoding occurs.

III. Off-line problem

A. Commodities

We define a number of commodities, each with its own conceptual "queue", at each node i:

- U_i^{cv}: uncoded session c packets also stored at node v
- $P_i^{\{c,c'\}j}$: joint poison packets from sessions c, c' coded at node j meant for both sinks
- $P_i^{cc'j}$: individual poison packets from sessions c, c' coded at node j meant for sink d_c
- $R_i^{cc'j}$: remedy for session c packets that has been coded with session c' packets at node j

B. Modified problem

Off-line, we can reduce complexity by considering a modified problem which reverses the direction of the poison flows while not changing the link capacity usage. Thus, any solution of the modified problem can be translated to a solution of the original problem and vice versa.

In the modified problem, coding two uncoded packets p_1, p_2 together produces two remedy packets, each at some node previously traversed by p_1, p_2 respectively. Each remedy packet is transformed by a *decoding* operation at some node into an uncoded packet and an individual poison packet. Two individual poison packets can be transformed by a *branching* operation into a joint poison packet, and all poison packets are removed at their original coding node, as shown in Figure 1 and described formally below.

For each link (a, b) such that $C_{ab} > \sum_c r_c$, set C_{ab} to $\sum_c r_c$, and let $\bar{C}_a$ be the larger of the total incoming capacity and total outgoing capacity of each node $a \in \mathcal{N}$. At each source node s_c we define two additional queues: a source queue U^c and an overflow queue $\bar{U}^c$. We also define a number of virtual links: at each source node s_c a virtual source link of capacity $\bar{C}_{s_c}$ from U^c to $U_{s_c}^{cs_c}$, and at each node a a virtual coding link, a virtual decoding link and a virtual branching link, each of capacity $\bar{C}_a/2$. Each real and virtual link e is associated with a transmission set $\mathcal{P}_e$ of pairs $(\mathcal{O}, \mathcal{D})$ such that packets from each queue in the set $\mathcal{O}$ are transformed into an equivalent number of packets in each queue in the set $\mathcal{D}$ via e:

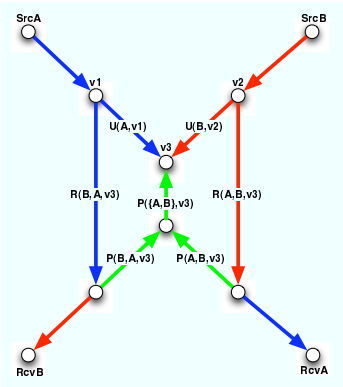

Fig. 1. Illustration of commodities in off-line case. The direction of the poison flows (labeled P) is the reverse of their physical (causal) direction from the canonical butterfly example.

- if e is the virtual source link for session c, $\mathcal{P}_e = (U^c, U_{s_c}^{cs_c})$
- if e is a real link $(a, b) \in \mathcal{L}$, $\mathcal{P}_e = \{(U_a^{cv}, U_b^{cv}), (U_a^{cv}, U_b^{ca}), (P_b^{\{c,c'\}j}, P_a^{\{c,c'\}j}), (P_b^{cc'j}, P_a^{cc'j}), (R_a^{cc'j}, R_b^{cc'j})\}$
- if e is the virtual coding link at node a, $\mathcal{P}_e = \left\{\left(\{U_a^{cv}, U_a^{c'v'}\}, \{R_{v'}^{cc'a}, R_v^{c'ca}\}\right) : c \neq c'; a \neq v, v'\right\}$
- if e is the virtual decoding link at node a, $\mathcal{P}_e = \left\{\left(R_a^{cc'j}, \{P_a^{cc'j}, U_a^{cj}\}\right) : c \neq c'; a \neq j\right\}$
- if e is the virtual branching link at a, $\mathcal{P}_e = \left\{\left(\{P_a^{cc'j}, P_a^{c'cj}\}, P_a^{\{c,c'\}j}\right) : c \neq c'; a \neq j\right\}$

Packets are removed from queues $U_{d_c}^{cv}$, $P_j^{c'cj}$, and $P_j^{\{c,c'\}j}$.

A solution is specified by giving the average transmission rate across each pair $(\mathcal{O}, \mathcal{D}) \in \mathcal{P}_e,\ e \in \mathcal{L}$.

C. Wireless case

Wireless broadcast links are denoted (a, Z), where a is the originating node and Z is the set of destination nodes. The network connectivity and link transmission rates depend on the transmitted signal and interference powers according to some underlying physical layer model. For simplicity, we consider a finite set $\mathcal{U}$ of sets of simultaneously achievable link rates (transmission scenarios). We denote by $C_{(a,Z),u}$ the capacity of link (a, Z) in set $u \in \mathcal{U}$. A solution to the multiple unicasts problem consists of a convex combination of sets in $\mathcal{U}$, which gives a set of average link capacities achievable by timesharing, and a network code that operates over the network with these average link capacities.

In the wireless case, the virtual links are defined exactly as in the wired case. A branching operation can also occur over a real wireless link, taking advantage of the broadcast medium. For real wireless links (a, Z), $\mathcal{P}_{(a,Z)} = \{(U_a^{cv}, U_b^{cv}), (U_a^{cv}, U_b^{ca}), (P_b^{\{c,c'\}j}, P_a^{\{c,c'\}j}), (P_b^{cc'j}, P_a^{cc'j}), (R_a^{cc'j}, R_b^{cc'j}), (U_a^{cv}, U_b^{cb'}), (\{P_b^{cc'j}, P_{b'}^{c'cj}\}, P_a^{\{c,c'\}j}) : c \neq c'; b \neq b'; b, b' \in Z\}$

IV. Off-line algorithm

We consider all queues $U_a^{cv}, U^c, R_a^{cc'v}, P_a^{cc'j}$ to be associated with session c. We consider each joint poison queue

$P_j^{\{c,c'\}j}$ as a pair of queues, one associated with each session c and c', such that any amount added to (removed from) $P_j^{\{c,c'\}j}$ means that the corresponding amount is added to (removed from) each of the pair of queues. All flows of packets occur between queues of the same session.

The algorithm is defined in terms of two parameters H and L, which can be interpreted in terms of a flow solution as follows. Running the algorithm for a particular choice of H and L yields a solution if there exists any (pairwise XOR-coded) flow solution in which each elementary flow has at most H real links and undergoes coding (and decoding) at most L times. Note that $H \geq L$ in any solution; the complexity of finding suitable values for H and L is discussed in Section V-E.

The overflow queue for each session c has a potential $\alpha_c l e^{\alpha_c B r_c}$ that is a function of its length l, where

$$\alpha_c = \frac{\epsilon}{8Fr_c} \quad (1)$$

$$F = 2H + 7L + 2, \quad (2)$$

and B is a constant that will be defined later. We divide each other queue into subqueues, one for each link for which it is an origin or destination. In the wireless case, we assume that each node broadcasts one message at a time and receives one message at a time, so for each queue only one origin subqueue and one destination subqueue are associated with (any number of) real links. Each subqueue Q has a potential $\phi_{c(Q)}(l_Q) = e^{\alpha_{c(Q)} l_Q}$ that depends on its length l_Q and its session $c(Q)$. For notational simplicity, we will abbreviate subscripts $c(Q)$ as subscripts Q.

Flow entering or leaving subqueues associated with session c is partitioned into blocks of size $b_c = (1+\epsilon)r_c$. Besides its true length l_Q, each subqueue Q has an approximate length $\tilde{l}_Q$ that is an integer multiple of the block size. The approximate length of a subqueue is updated only when its true length has changed by at least one block since the last update of its approximate length, as follows: $\tilde{l}_Q$ is set to kb_Q where $k = \left\lfloor \frac{l_Q}{b_Q} \right\rfloor - 1$ if Q is an origin subqueue or $k = \left\lceil \frac{l_Q}{b_Q} \right\rceil + 1$ if Q is a destination subqueue. Between updates, l_Q and $\tilde{l}_Q$ satisfy $l_Q - 3b_Q \leq \tilde{l}_Q \leq l_Q$ for an origin subqueue, or $l_Q \leq \tilde{l}_Q \leq l_Q + 3b_Q$ for a destination subqueue.

We denote by $\bar{\mathcal{P}}_e$ the subset of $\mathcal{P}_e$ consisting of pairs $(\mathcal{O}, \mathcal{D}) \in \mathcal{P}_e$ satisfying

$$\min_{Q \in \mathcal{O}} \tilde{l}_Q > 0$$

$$\max_{Q \in \mathcal{D}} \tilde{l}_Q < Br_Q + \ln((L+1)\rho)/\alpha_Q + 3b_Q,$$

where $\rho = \max_c r_c / \min_c r_c$.

A. Wired case

In each round t, the algorithm carries out the following:

1) Add $(1+\epsilon)r_c$ units to the overflow queue $\bar{U}^c$ of each session c, then transfer as much as possible to U^c subject to a maximum length constraint of Br_c for U^c
2) For each real and virtual link e, flow is pushed for zero or more origin-destination pairs $(\mathcal{O}, \mathcal{D}) \in \bar{\mathcal{P}}_e$ such that the total amount pushed is at most the link capacity C_e. Specifically, initialize C to C_e and repeat
   - Choose the pair $(\mathcal{O}, \mathcal{D}) \in \bar{\mathcal{P}}_e$ that maximizes
   $$w_{(\mathcal{O},\mathcal{D})} = \sum_{Q \in \mathcal{O}} \phi_c'(\tilde{l}_Q) - \sum_{Q \in \mathcal{D}} \phi_c'(\tilde{l}_Q). \quad (3)$$
   Let
   $$C' = \min\left(C, \min_{Q \in (\mathcal{O} \cup \mathcal{D})} (b_Q - |\delta_Q|)\right)$$
   where δ_Q is the change in l_Q since the last update of $\tilde{l}_Q$. Subtract C' units from C, subtract C' units from l_Q for each $Q \in \mathcal{O}$ and add units C' to l_Q for each $Q \in \mathcal{D}$. For $Q \in (\mathcal{O} \cup \mathcal{D})$, if $C' = b_Q - |\delta_Q|$, update $\tilde{l}_Q$.
   - If $C = 0$ then end.
3) Zero out all subqueues $U_{d_c}^{cv}$, $P_j^{c'cj}$ and $P_j^{\{c,c'\}j}$.
4) For each queue that has at least one subqueue whose actual length has changed during the round, reallocate packets to equalize the actual lengths of all its subqueues. If the actual length of any subqueue has changed by at least one block since the last update of its approximate length, update its approximate length.

When the amount of flow remaining in the network queues is an ϵ-fraction of the total amount that has entered the network, the flow values for each link are averaged over all rounds to give the solution.

B. Wireless case

The algorithm is the same as for the wired case, except for Phase 2, which is as follows. For each virtual link e, flow is pushed for zero or more origin-destination pairs $(\mathcal{O}, \mathcal{D}) \in \bar{\mathcal{P}}_e$ exactly as in the wired case. For each real link $e = (a, Z)$, flow is pushed for zero or more origin-destination pairs $(\mathcal{O}, \mathcal{D}) \in \bar{\mathcal{P}}_e$ such that the total amount pushed is at most the average link capacity $\lambda_u C_{e,u}$ for some λ_u satisfying $\sum_{u \in \mathcal{U}} \lambda_u \leq 1$.

Specifically, initialize T to 1 and repeat

- For each real link $e = (a, Z)$, let
$$w_e = \max_{(\mathcal{O},\mathcal{D}) \in \bar{\mathcal{P}}_e} w_{(\mathcal{O},\mathcal{D})}$$
$$(\mathcal{O}_e, \mathcal{D}_e) = \arg\max_{(\mathcal{O},\mathcal{D}) \in \bar{\mathcal{P}}_e} w_{(\mathcal{O},\mathcal{D})}$$
where $w_{(\mathcal{O},\mathcal{D})}$ is defined in (3). Choose the set $u \in \mathcal{U}$ which maximizes
$$y_u = \sum_e w_e C_{e,u}. \quad (4)$$
Let
$$T' = \min\left(T, \min_{e:\, C_{e,u} > 0} \min_{Q \in (\mathcal{O}_e \cup \mathcal{D}_e)} \frac{b_Q - |\delta_Q|}{C_{e,u}}\right)$$
where δ_Q is the change in l_Q since the last update of $\tilde{l}_Q$. Subtract T' from T, and for each e, subtract $T' C_{e,u}$ units from l_Q for each $Q \in \mathcal{O}_e$ and add the

same amount to l_Q for each $Q \in \mathcal{D}_e$, updating $\tilde{l}_Q$ if $T'C_{e,u} = b_Q - |\delta_Q|$.

- If $T = 0$ then end.

Theorem 1 (Wired networks): If input rates $(r_c(1+2\epsilon))$ are achievable with pairwise XOR coding for some $\epsilon > 0$, then the algorithm obtains a solution that achieves rates (r_c) in time

$$O\left(\frac{N^3MK^2H\log H\log(NK)}{\epsilon^2}\left(KL+\ln\left(\frac{1}{\epsilon}\right)\right)\right).$$

Theorem 2 (Wireless networks): If input rates $(r_c(1+2\epsilon))$ are achievable with pairwise XOR coding for some $\epsilon > 0$, then the algorithm obtains a solution that achieves rates (r_c) in time

$$O\left(\frac{N^3KH\log H(NK\log(NK)+|\mathcal{U}|)}{\epsilon^2}\left(KL+\ln\left(\frac{1}{\epsilon}\right)\right)\right).$$

The proof is given in the following section.

V. Potential analysis

A. Flow solution-based algorithm

Our analysis builds on the approach in [2]. We denote by $Q(t)$ the actual length of a subqueue Q at the end of Phase 1 of round t. From [2], the increase in potential during Phase 1 is upper bounded by

$$\sum_c (1+\epsilon) r_c \phi_c'(U^c(t)). \tag{5}$$

We lower bound the decrease in potential during Phases 2 and 3 by comparison with the potential decrease resulting from pushing flow based on a flow solution for rates $f_c = (1+2\epsilon)r_c$. This algorithm differs from the back pressure algorithm only in the portion of Phase 2 determining the amount of flow pushed for each pair $(\mathcal{O},\mathcal{D}) \in \bar{\mathcal{P}}_e$ of each link e.

Consider a flow solution for rates f_c. Partition the flow for each session c into elementary flows $\mathcal{F}_n^c$ whose size is denoted f_n^c. We say that a subqueue is in an elementary flow $\mathcal{F}_n^c$ if packets from that flow are transferred into or out of the subqueue. Note that each flow $\mathcal{F}_n^c$ starts from queue U^c, which consists of a single subqueue, and that $f_c = \sum_n f_n^c$. A subqueue in $\mathcal{F}_n^c$ is considered upstream or downstream of another according to the direction of flow in the modified problem defined in Section III-B. An elementary flow may be truncated by removing from it all links and subqueues upstream or downstream of some subqueue in the flow. Two such flows are said to share a path segment if they are coded together in the flow solution and contain at least one real or virtual link in common.

Phase 2 of the flow solution-based algorithm consists of a preprocessing part and a flow pushing part.

1) Preprocessing procedure:

- Initialization:
  Remove from each $\mathcal{F}_n^c$ any portions that are downstream of a subqueue Q in $\mathcal{F}_n^c$ for which $l_Q \le 3b_c$. Note that all subqueues of each queue have been equalized in Phase 4 of the previous round. Let $\mathcal{G}$ be the set of flows $\mathcal{F}_n^c$ containing some subqueue of length at least $Br_c + \ln((L+1)\rho)/\alpha_c$. For flows $\mathcal{F}_n^c \in \mathcal{G}$, let $\bar{Q}_n^c$ be the furthest downstream origin subqueue in $\mathcal{F}_n^c$ of length at least $Br_c + \ln((L+1)\rho)/\alpha_c$; for flows $\mathcal{F}_n^c \notin \mathcal{G}$, let $\bar{Q}_n^c$ be the longest origin subqueue in $\mathcal{F}_n^c$ (if there is a tie, choose the furthest downstream).
  Initialize $\mathcal{H}$ as the set $\mathcal{H}_0$ of all flows $\mathcal{F}_n^{*c}$, initialize $\mathcal{H}_1$ and $\mathcal{H}_2$ as empty sets, and for each c, n, initialize Q_n^{*c} as $\bar{Q}_n^c$.
- Phase A: Repeat
  - Choose some flow $\mathcal{F}_n^c \in \mathcal{G} \cap \mathcal{H}$ and remove from $\mathcal{H}$ all flows $\mathcal{F}_{n'}^{c'}$ such that $\mathcal{F}_{n'}^{c'}$ shares a path segment with $\mathcal{F}_n^c$. Remove $\mathcal{F}_n^c$ from $\mathcal{H}$ and add it to $\mathcal{H}_1$.
  - If there is no such flow remaining, end Phase A.
- Phase B: Associate with each flow $\mathcal{F}_n^c \in \mathcal{H}$ a weight w_n^c, initially set to $\phi_c'(U^c(t))$. For each flow $\mathcal{F}_{n'}^{c'} \in \mathcal{H}$, set $\mathcal{I}_{n'}^{c'}$ to be the set of subqueues $\bar{Q}_n^c$ such that $\mathcal{F}_{n'}^{c'}$ is the shared flow corresponding to $\bar{Q}_n^c$, and either
  - $\bar{Q}_n^c$ is a remedy or individual poison subqueue whose corresponding branching link is in $\mathcal{F}_n^c$ and whose corresponding coding link is in $\mathcal{F}_{n'}^{c'}$, or
  - $\bar{Q}_n^c$ is a joint poison subqueue whose corresponding branching link is in $\mathcal{F}_{n'}^{c'}$.

  Repeat
  - Choose some flow $\mathcal{F}_{n'}^{c'} \in \mathcal{H}$ such that
  $$w_{n'}^{c'} \le \sum_{\bar{Q}_n^c \in \mathcal{I}_{n'}^{c'}} \left(\phi_c'(\bar{Q}_n^c(t)) - w_n^c\right), \tag{6}$$
    - remove $\mathcal{F}_{n'}^{c'}$ from $\mathcal{H}$,
    - remove any subqueues in $\mathcal{F}_{n'}^{c'}$ from $\mathcal{I}_n^c \ \forall\, c, n$,
    - set the weight w_n^c of each flow $\mathcal{F}_n^c \in \mathcal{I}_{n'}^{c'}$ to $\phi_c'(\bar{Q}_n^c(t))$.
  - If there is no such flow, for each $\mathcal{I}_{n'}^{c'}$ and each $Q_n^{*c} \in \mathcal{I}_{n'}^{c'}$ set Q_n^{*c} to U^c, and end Phase B.
- Phase C: Repeat
  - Choose some flow $\mathcal{F}_n^c \in \mathcal{H}$ such that the longest individual poison subqueue along one of its poison segments, Q, is longer than Q_n^{*c}, and the entire joint poison portion of that segment together with the branching link are not in the corresponding shared flow. Set Q_n^{*c} to Q and remove from $\mathcal{F}_n^c$ all but the portion downstream of Q.
  - If there is no such flow remaining, end Phase C.

Observe that:

- At most L flows are removed by each flow in $\mathcal{G}$, and
$$\phi_c'(Br_c + \ln((L+1)\rho)/\alpha_c) \ge (L+1)\max_c \phi_c'(Br_c)$$
$$\ge (L+1)\max_c \phi_c'(U^c(t) \tag{7}$$
- $f_n^c = f_{n'}^{c'}$ for all $\mathcal{F}_n^c$ and $\mathcal{F}_{n'}^{c'}$ that share a path segment.
- At the end of the preprocessing procedure,
$$2\sum_{\mathcal{F}_n^c \in \mathcal{H}} \phi_c'(Q_n^{*c}(t)) f_n^c \ge \sum_{\mathcal{F}_n^c \in \mathcal{H}} \phi_c'(\bar{Q}_n^c(t)) f_n^c. \tag{8}$$
  To see this, note that at the end of Phase B there is a one-to-one correspondence between flows $\mathcal{F}_n^c \in \mathcal{H}$ for

which $Q_n^{*c} \neq \bar{Q}_n^c$ and elements $\bar{Q}_n^c$ in the sets $\mathcal{I}_{n'}^{c'}$ of flows $\mathcal{F}_{n'}^{c'} \in \mathcal{H}$. For each $\mathcal{F}_{n'}^{c'} \in \mathcal{H}$, by (6),

$$w_{n'}^{c'} > \sum_{\bar{Q}_n^c \in \mathcal{I}_{n'}^{c'}} \left(\phi_c'(\bar{Q}_n^c(t)) - w_n^c\right).$$

Multiplying by f_n^c and summing over all $\mathcal{F}_{n'}^{c'} \in \mathcal{H}$, and noting that at the end of the preprocessing procedure

$$\phi_c'(Q_n^{*c}(t)) \geq w_n^c \tag{9}$$

gives (8).

- Phase B and C maintain the invariant

$$\sum_{\mathcal{F}_n^c \in \mathcal{H}} w_n^c f_n^c \geq \sum_{\mathcal{F}_n^c \in \mathcal{H}_2} \phi_c'(U^c(t)) f_n^c \tag{10}$$

where $\mathcal{H}_2$ is the value of $\mathcal{H}$ at the start of Phase B. The invariant holds since both sides are equal at the start of Phase B, and the left-hand side is monotonically non-decreasing. From (7), (9) and (10), at the end of the preprocessing procedure we have

$$\sum_{\mathcal{F}_n^c \in (\mathcal{H}_1 \cup \mathcal{H})} \phi_c'(Q_n^{*c}(t)) f_n^c \geq \sum_{\mathcal{F}_n^c \in \mathcal{H}_0} \phi_c'(U^c(t)) f_n^c. \tag{11}$$

2) Flow pushing procedure: For each flow $\mathcal{F}_n^c \in \mathcal{H}_1 \cup \mathcal{H}$, let $\mathcal{F}_n^{*c}$ be the portion of $\mathcal{F}_n^c$ downstream of Q_n^{*c}. Partition its links into a set $\mathcal{G}_n^{*c}$ of subsets. Each of these subsets $\mathcal{S} \in \mathcal{G}_n^{*c}$ may be

- a path from Q_n^{*c}, if it is a poison subqueue, to its associated coding node $a(\mathcal{S})$
- a path from Q_n^{*c}, if it is a remedy subqueue, up to and including its associated decoding link at node $w(\mathcal{S})$, and the associated poison path from $w(\mathcal{S})$ to its associated coding node $a(\mathcal{S})$ via the branching link at node $b(\mathcal{S})$,
- a path associated with uncoded flow ending in sink node d_c, or
- a path associated with uncoded flow ending in a coding link at a node $a(\mathcal{S})$, together with the associated remedy path up to and including the decoding link at a node $w(\mathcal{S})$, and the associated poison path from w to a via the branching link at a node b.

Note that each subset starts either at Q_n^{*c} or at an uncoded subqueue.

The flow solution-based algorithm pushes flow as follows. First, for each pair $(\mathcal{S}_c, \mathcal{S}_{c'}) \in \mathcal{G}_n^{*c} \times \mathcal{G}_{n'}^{*c'}$ that shares a path segment, note that $f_n^c = f_{n'}^{c'}$ and that Phase B of the preprocessing procedure ensures that both subsets or neither contains the coding link; in the latter case both or neither contains the branching link. Thus, the following are the only two cases:

- Case 1: One of the subsets, say $\mathcal{S}_c$, contains both the coding link at $a(\mathcal{S}_c)$ and the branching link at $b(\mathcal{S}_c)$, while $\mathcal{S}_{c'}$ contains the coding link but not the branching link. Phase C of the preprocessing procedure ensures that all session c individual poison subqueues along the joint poison path segment are shorter than Q_n^{*c}. Push f_n^c units through $\mathcal{S}_c \cup \mathcal{S}_{c'}$, pushing session c individual poison units through the joint poison path segment.
- Case 2: Both subsets contain the same portion of the joint poison path segment. Push f_n^c units through $\mathcal{S}_c \cup \mathcal{S}_{c'}$.

Next, for each subset $\mathcal{S}$ of some $\mathcal{F}_n^{*c}$ that does not share any path segment with $\mathcal{F}_{n'}^{*c'}$ for all $c' \neq c, n'$, we have the following cases:

- Case 1: Q_n^{*c} is an individual poison subqueue in $\mathcal{S}$. Phase B of the preprocessing procedure ensures that Q_n^{*c} is the longest individual poison subqueue in $\mathcal{F}_{n'}^{*c'}$. Push f_n^c individual poison units through $\mathcal{S}$.
- Case 2: Q_n^{*c} is a joint poison subqueue in $\mathcal{S}$. Push f_n^c joint poison units along the path in $\mathcal{S}$ downstream of Q_n^{*c}.
- Case 3: Q_n^{*c} is a remedy subqueue in $\mathcal{S}$. Phase C of the preprocessing procedure ensures that Q_n^{*c} is longer than all session c individual poison subqueues along the primary path of $\mathcal{S}$. Push f_n^c remedy units along $\mathcal{S}$ through the decoding link at $w(\mathcal{S})$, and f_n^c individual poison units along the primary path of $\mathcal{S}$.
- Case 4: Q_n^{*c} is an uncoded subqueue or is not in $\mathcal{S}$. Push f_n^c uncoded units along the primary path of $\mathcal{S}$ starting from its longest session c uncoded subqueue.

Note that flow is pushed only from origin subqueues Q for which $l_Q > 3b_Q \Rightarrow \tilde{l}_Q > 0$, and only to destination subqueues Q for which $l_Q < Br_Q + \ln((L+1)\rho)/\alpha_Q \Rightarrow \tilde{l}_Q < Br_Q + \ln((L+1)\rho)/\alpha_Q + 3b_Q$.

B. Potential decrease in Phases 2 and 3

The decrease in potential from pushing f units across a link from a set $\mathcal{O}$ of origin to a set $\mathcal{D}$ of destination subqueues is at least

$$\sum_{Q \in \mathcal{O}} \left(f\phi_Q'(l_Q) - f^2\phi_Q''(l_Q)\right) - \sum_{Q \in \mathcal{D}} \left(f\phi_Q'(l_Q) + f^2\phi_Q''(l_Q + f)\right)$$

where l_Q denotes the initial length of each subqueue Q.

1) Flow solution-based algorithm: We denote by $\mathcal{O}(\mathcal{F}_n^{*c})$ and $\mathcal{D}(\mathcal{F}_n^{*c})$ the sets of origin and destination subqueues of a flow $\mathcal{F}_n^{*c}$, and by $\mathcal{O}_{c,e}$ and $\mathcal{D}_{c,e}$ the sets of session c origin and destination subqueues of a link e. For each subqueue Q, denote by f_Q the total flow out of Q (if Q is an origin subqueue) or into Q (if Q is a destination subqueue) in the flow pushing procedure of the flow-solution based algorithm. The potential drop over Phases 2 and 3 in the flow solution-based algorithm is at least

$$\begin{aligned}
&\sum_{c,e} \left(\sum_{Q \in \mathcal{O}_{c,e}} \left(f_Q\phi_c'(l_Q) - f_Q^2\phi_c''(l_Q + f_Q)\right) \right. \\
&\qquad \left. - \sum_{Q \in \mathcal{D}_{c,e}} f_Q \left(\phi_c'(l_Q) + f_Q^2\phi_c''(l_Q + f_Q)\right) \right) \\
&\geq \sum_{\mathcal{F}_n^{*c}} \left(\sum_{Q \in \mathcal{O}(\mathcal{F}_n^{*c})} f_n^c \left(\phi_c'(l_Q) - f_c\phi_c''(l_Q + f_c)\right) \right.
\end{aligned}$$

$$- \sum_{Q \in \mathcal{D}(\mathcal{F}_n^{*c})} f_n^c \left(\phi_c'(l_Q) + f_c \phi_c''(l_Q + f_c)\right)\Bigg)$$

Since all subqueues of each queue have been equalized in Phase 4 of the previous round, and since each flow $\mathcal{F}_n^{*c}$ has one origin subqueue of length $Q_n^{*c}(t)$, at most $L+1$ destination subqueues each of length less than $3b_c$, and a total of at most F subqueues, this potential drop is lower-bounded by

$$\begin{aligned}
&\sum_{\mathcal{F}_n^c \in (\mathcal{H}_1 \cup \mathcal{H})} f_n^c \left(\phi_c'(Q_n^{*c}(t)) - (L+1)\,\phi_c'(3b_c)\right) \\
&- \sum_{\mathcal{F}_n^c \in \mathcal{H}_1} F f_n^c f_c \phi_c''(Q_n^{*c}(t) + f_c) \\
&- \sum_{\mathcal{F}_n^c \in \mathcal{H}} F f_n^c f_c \phi_c''(\bar{Q}_n^c(t) + f_c)
\end{aligned}$$

From (1), we have

$$\begin{aligned}
F f_c \phi_c''(l + f_c) &\leq F(1+2\epsilon) r_c \phi_c''(l + (1+2\epsilon) r_c) \\
&= F(1+2\epsilon) r_c \alpha_c^2 e^{\alpha_c l + \alpha_c (1+2\epsilon) r_c} \\
&= F(1+2\epsilon) r_c \alpha_c e^{\alpha_c (1+2\epsilon) r_c} \phi_c'(l) \\
&= \frac{\epsilon(1+2\epsilon)}{8} e^{\frac{\epsilon(1+2\epsilon)}{8F}} \phi_c'(l) \\
&\leq \epsilon \phi_c'(l)/4. \qquad (12)
\end{aligned}$$

This yields, using (8), the following lower bound on the potential drop:

$$\begin{aligned}
&\sum_{\mathcal{F}_n^c \in (\mathcal{H}_1 \cup \mathcal{H})} f_n^c \left(\phi_c'(Q_n^{*c}(t)) - (L+1)\,\phi_c'(3b_c)\right) \\
&\quad - \sum_{\mathcal{F}_n^c \in \mathcal{H}_1} f_n^c \epsilon \phi_c'(Q_n^{*c}(t))/4 - \sum_{\mathcal{F}_n^c \in \mathcal{H}} f_n^c \epsilon \phi_c'(\bar{Q}_n^c(t))/4 \\
&\geq \sum_c r_c \left(\left(1 + \frac{3\epsilon}{2} - \epsilon^2\right) \phi_c'(U^c(t)) \right. \\
&\quad \left. - (L+1)(1+2\epsilon)\phi_c'(3b_c)\right) \qquad (13)
\end{aligned}$$

2) Back pressure algorithm: If the block size b_c were infinitesimally small rather than $(1+\epsilon)r_c$, then $l_Q = \tilde{l}_Q$ and the procedure in Phase 2 of the back pressure algorithm would give, for each link e, the maximum possible potential decrease from pushing flow for zero or more origin-destination pairs $(\mathcal{O}, \mathcal{D}) \in \bar{\mathcal{P}}_e$ such that the total amount pushed is at most the link capacity C_e. Since $b_c = \Theta(r_c)$ and

$$\phi_c'(l + \Theta(r_c)) - \phi_c'(l) \leq \Theta(r_c)\phi_c''(l + \Theta(r_c)),$$

the back pressure algorithm achieves, for each link e, a potential decrease of at least that achieved by the flow solution-based algorithm minus an error term

$$\sum_{Q \in (\mathcal{O}_e \cup \mathcal{D}_e)} \Theta(f_Q r_Q) \phi_Q''(l_Q + \Theta(r_Q))$$

where f_Q is the amount of flow added or removed from Q in Phase 2 of the flow solution-based algorithm. Since $f_Q = \Theta(r_Q)$, decreasing each α_c by some constant factor ensures that (13) applies to the back pressure algorithm.

C. Overall potential change and number of rounds

The potential does not increase during Phase 4. Thus, from (5) and (13), the overall potential decrease during the round is lower bounded by

$$\begin{aligned}
&\sum_c r_c \left(\left(\frac{\epsilon}{2} - \epsilon^2\right) \phi_c'(U^c(t)) - (L+1)(1+2\epsilon)\phi_c'(3b_c)\right) \\
&= \sum_c r_c \left(\frac{\epsilon}{2} - \epsilon^2\right) \phi_c'(U^c(t)) \\
&\quad - (L+1)(1+2\epsilon)\frac{\epsilon}{8F} e^{\frac{3\epsilon(1+\epsilon)}{8F}}
\end{aligned}$$

If $U^c(t) = Br_c$ for some c, then the decrease in potential is at least

$$\begin{aligned}
&r_c \left(\frac{\epsilon}{2} - \epsilon^2\right) \phi_c'(Br_c) - K(L+1)(1+2\epsilon)\frac{\epsilon}{8F} e^{\frac{3\epsilon(1+\epsilon)}{8F}} \\
&= r_c \alpha_c \left(\frac{\epsilon}{2} - \epsilon^2\right) e^{\alpha_c B r_c} - K(L+1)(1+2\epsilon)\frac{\epsilon}{8F} e^{\frac{3\epsilon(1+\epsilon)}{8F}}
\end{aligned}$$

which is non-negative if

$$\begin{aligned}
Br_c &= \frac{1}{\alpha_c} \ln\left(\frac{K(L+1)(1+2\epsilon)}{\epsilon(1-2\epsilon)}\right) + 3b_c \\
&= \Theta\left(\frac{1}{\alpha_c} \ln\left(\frac{KL}{\epsilon}\right)\right)
\end{aligned}$$

If $U^c(t) < Br_c$ for all c, the overflow queues are empty and, since there are $\Theta(NMK)$ session-c subqueues (in the wired case, or $\Theta(N^2K)$ in the wireless case) each of potential less than

$$\phi_c(Br_c + \ln((L+1)\rho)/\alpha_c + 4b_c) = \Theta\left(L\rho e^{\frac{\epsilon B}{8F}}\right),$$

the overall potential in the system at the end of the round is at most

$$\Theta\left(NMK^2 L\rho e^{\frac{\epsilon B}{8F}}\right)$$

By induction, this is also an upper bound on the total potential at the end of every round. The length of the overflow queue for session c is thus never more than

$$O\left(\frac{NMK^2 L\rho e^{\frac{\epsilon B}{8F}}}{\alpha_c e^{\frac{\epsilon B}{8F}}}\right) = O\left(\frac{NMK^2 L\rho}{\alpha_c}\right)$$

and the total units for session c is at most

$$\begin{aligned}
&O\left(\frac{NMK^2 L\rho}{\alpha_c} + NMK(Br_c + \ln(L\rho)/\alpha_c + 4b_c)\right) \\
&= O\left(\frac{NMKHr_c}{\epsilon}\left(KL\rho + \log\left(\frac{1}{\epsilon}\right)\right)\right)
\end{aligned}$$

Thus, at most

$$O\left(\frac{NMKH}{\epsilon}\left(KL\rho + \log\left(\frac{1}{\epsilon}\right)\right)\right)$$

rounds of input flow for each session remain in the network at any time. For

$$t = O\left(\frac{NMKH}{\epsilon^2}\left(KL\rho + \log\left(\frac{1}{\epsilon}\right)\right)\right)$$

the amount for each session remaining in the network is at most a fraction ϵ of the total amount that has entered the network up to round t.

D. Number of operations

In each round, at each node a, pushing flow across links results in a total decrease of at most $O(\bar{C}_a)$ in the actual lengths of origin subqueues and a total increase of at most $O(\bar{C}_a)$ in the actual lengths of destination subqueues. The total change in subqueue lengths from rebalancing is not more than the total change resulting from pushing flow across links. Thus, at most $O(ND)$ approximate subqueue lengths in the network are updated in each round, where $D = \sum_{a\in\mathcal{N}} \bar{C}_a/(N \min_c r_c)$. Assuming the subqueue differences computed by the algorithm can be stored, only those differences involving subqueues whose approximate lengths have changed are recomputed.

For each real and virtual link e, the values of $w_{(\mathcal{O},\mathcal{D})}$, defined in (3), for pairs $(\mathcal{O},\mathcal{D}) \in \mathcal{P}_e$ are stored in a sorted list. For a coding link, this list has length $O(N^2K^2)$, and a change to an uncoded subqueue's approximate length affects NK entries, requiring $O(NK\log(NK))$ operations. This is the worst case complexity for updating any approximate subqueue length.

Thus in the wired case each round has complexity $O(N^2KD\log(NK))$, and the algorithm has complexity

$$O\left(\frac{N^3MK^2HD\log(NK)}{\epsilon^2}\left(KL\rho+\log\left(\frac{1}{\epsilon}\right)\right)\right). \quad (14)$$

In the wireless case, assuming that for each wireless link (a,Z), $|Z| = O(1)$, updating any subqueue's approximate length similarly requires at most $O(NK\log(NK))$ operations. A further $O(|\mathcal{U}|)$ operations suffices to update the values of y_u and find the maximum among them.

Thus, each round has complexity $O(ND(NK\log(NK)+|\mathcal{U}|)$, and the algorithm has complexity

$$O\left(\frac{N^3KHD(NK\log(NK)+|\mathcal{U}|)}{\epsilon^2}\left(KL\rho+\log\left(\frac{1}{\epsilon}\right)\right)\right). \quad (15)$$

E. Choosing parameters H and L

Appropriate values for parameters H and L can be found as follows. Let $\underline{H}$ be the maximum over all sessions of the minimum source-receiver path length for a session, and $\overline{H}$ and $\overline{L}$ upper bounds on values of interest for H and L respectively[3]. Define $\underline{L} = \log(1/\epsilon)/K\rho$, $\underline{h} = \lceil \log_2 \underline{H} \rceil$ and $\underline{l} = \lfloor \log_2 \underline{L} \rfloor$. We run the algorithm for the following sequence of values

$$\begin{aligned}(H,L) = {} & (2^{\underline{h}},2^{\underline{h}}),(2^{\underline{h}+1},2^{\underline{h}+1}),\ldots,(2^{\underline{l}},2^{\underline{l}}),(2^{\underline{l}+1},2^{\underline{l}}),\\ & (2^{\underline{l}+2},2^{\underline{l}}),(2^{\underline{l}+1},2^{\underline{l}+1}),(2^{\underline{l}+3},2^{\underline{l}}),(2^{\underline{l}+2},2^{\underline{l}+1}),\\ & (2^{\underline{l}+4},2^{\underline{l}}),\ldots,(2^{\underline{l}+2},2^{\underline{l}+2}),\ldots,\\ & (2^{\underline{l}+n},2^{\underline{l}}),\ldots,(2^{\underline{l}+n-\lfloor n/2\rfloor},2^{\underline{l}+\lfloor n/2\rfloor}),\ldots,(\overline{H},\overline{L})\end{aligned}$$

[3]In the worst-case, $\overline{H}$ and $\overline{L}$ can be $\Theta(n)$, e.g. for two source-sink pairs communicating in opposite directions from either end of a wireless network consisting of a line of nodes, but in general for large networks they are significantly smaller.

if $\underline{h} \le \underline{l}$, otherwise we use the sequence

$$\begin{aligned}(H,L) = {} & (2^{\underline{h}},2^{\underline{l}}),(2^{\underline{h}+1},2^{\underline{l}}),(2^{\underline{h}},2^{\underline{l}+1}),\\ & (2^{\underline{h}+2},2^{\underline{l}}),\ldots,(2^{\underline{h}+2-i_2},2^{\underline{l}+i_2}),\ldots,\\ & (2^{\underline{h}+n},2^{\underline{l}}),\ldots,(2^{\underline{h}+n-i_n},2^{\underline{l}+i_n}),\ldots,(\overline{H},\overline{L})\end{aligned}$$

where $i_n = \lfloor(\underline{h}+n-\underline{l})/2\rfloor$. From (14) and (15), the total time for finding a solution, if one exists, is a factor of $\Theta(\log H)$ higher than if the optimal values of H and L were known in advance.

VI. Dynamic online setting

Our off-line algorithm can be readily adapted to a dynamic online setting if we assume a separate low-rate channel for control information, i.e. queue length updates and requests for remedy packets. Unlike off-line where we could change flow directions, in the online case we are constrained by the actual directions of flow of packets. Thus, we do essentially the reverse of the off-line case: instead of having remedy generate poison en route to the coding node, in the online case we have poison generate remedy (remotely, by means of a remedy request) as it travels away from the coding node.

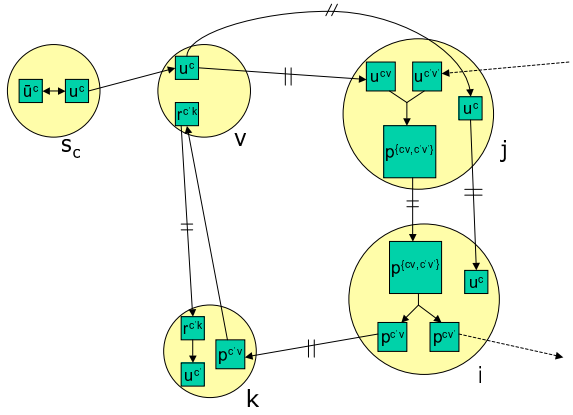

Fig. 2. Illustration of commodities in online case

We further consider two practical issues. The first is delay in transmission of control information. The effect of old information in queue length-based algorithms has been considered in a number of settings: infrequently updated or delayed queue length information can be detrimental in multi-server systems if arriving tasks choose the apparently least loaded server [10]; infrequently updated information does not affect asymptotic stability of back pressure multi-hop routing [11]; infrequently updated or delayed information does not affect asymptotic stability of joint scheduling and congestion control where each flow has a single specified route [16], [3]. We extend the results of [11] to back pressure network coding when control information is transmitted with delay T. Second, in distributed real-time applications the amount of computation at each node is often limited. Thus, our dynamic algorithm is essentially a simpler version of our off-line algorithm. It is able to stabilize all queues in the network for any stabilizable set of exogenous source rates,

but requires more rounds to converge to an $(1+\epsilon)$-optimal average solution. Computation can be further reduced by adapting the algorithm to optimize over a subset of coding options, which may be discovered by random exploration.

In the online version, we assume that each node has a total incoming capacity and a total outgoing capacity of at most μ. Instead of source queues U^c at the source nodes, we have "unaffiliated" uncoded queues U_i^c at each node i (in addition to uncoded queues U_i^{cv} that are "affiliated" with node v). For a coding link e at a node j, $\mathcal{P}_e = \{(\{U_j^{cv}, U_j^{c'v'}\}, P_j^{\{cv,c'v'\}})\}$; corresponding transmission rates are denoted $\gamma_j^{\{cv,c'v'\}}$.[4] For a branching link e at a node i, $\mathcal{P}_e = \{(P_i^{\{cv,c'v'\}}, \{P_i^{cv'}, P_j^{c'v}\})\}$; corresponding transmission rates are denoted $\sigma_i^{\{cv,c'v'\}}$. For a requesting link e at a node k, $\mathcal{P}_e = \{(P_k^{cv'}, R_{v'}^{ck})\}$; the corresponding transmission rates are denoted $\kappa_k^{v'}$ and the transformation is done by means of a remedy request sent from k to v' on the control channel. For a decoding link e at node k, $\mathcal{P}_e = \{(R_k^{ck}, U_k^c)\}$ and the corresponding transmission rates are denoted η_k^c. All virtual links have capacity $\mu/2$. For a real link $(a,b) \in \mathcal{L}$, $\mathcal{P}_{(a,b)}$ consists of pairs (U_a^c, U_b^c) (rate denoted ν_{ab}^c), pairs (U_a^{cv}, U_b^{cv}) (rate ν_{ab}^{cv}), pairs (U_a^c, U_b^{ca}) (rate χ_{ab}^c), pairs $\left(P_a^{\{cv,c'v'\}}, P_b^{\{cv,c'v'\}}\right)$ (rate $\pi_{ab}^{\{cv,c'v'\}}$), pairs $\left(P_a^{cj}, P_b^{cj}\right)$ (rate π_{ab}^{cj}), and pairs $\left(R_a^{cj}, R_b^{cj}\right)$ (rate ρ_{ab}^{cj}). For a wireless link (a,Z), the transmission set contains all pairs in $\mathcal{P}_{(a,b)}, b \in Z$ as well as pairs $(P_a^{\{cv,c'v'\}}, \{P_b^{cv'}, P_{b'}^{c'v}\})$ and $\left(U_a^c, U_b^{cb'}\right)$ where $b, b' \in Z$. Our subsequent development is for the wired case; extension to the wireless case is straightforward. We denote by $\bar{\mathcal{P}}_e$ the subset of $\mathcal{P}_e$ consisting of pairs $(\mathcal{O},\mathcal{D}) \in \mathcal{P}_e$ satisfying

$$\max_{Q\in\mathcal{D}} \tilde{l}_Q \quad < \quad 2V(L+1).$$

where V will be defined later. Queues and links are illustrated in Figure 2.

In the online case we do not divide queues into subqueues. We assume that control information is transmitted with a delay of less than T and that time is slotted with time slots of duration T. This allows a remedy transformation to occur within a time slot , and allows back pressure control decisions to be made based on queue length information from the start of the previous time slot. In the wired case, a decision is taken once every time slot to choose one pair $(\mathcal{O},\mathcal{D}) \in \bar{\mathcal{P}}_e$ to be served on each link e. In the wireless case, a transmission scenario is selected at the start of each time slot and if the channel states change.

In each time slot $(t, t+T]$, the following steps are carried out:

1) For each real and virtual link e, choose the pair $(\mathcal{O},\mathcal{D}) \in \bar{\mathcal{P}}_e$ that maximizes

$$w_{(\mathcal{O},\mathcal{D})} = \sum_{Q\in\mathcal{O}} \phi'(Q(t-T)^+) - \sum_{Q\in\mathcal{D}} \phi'(Q(t-T)^+)$$
$$= \sum_{Q\in\mathcal{O}} (Q(t-T)^+) - \sum_{Q\in\mathcal{D}} (Q(t-T)^+)$$

Transfer across the chosen $(\mathcal{O},\mathcal{D})$ pair, at the instantaneous rate of link e, up to the amount $\max_{Q\in\mathcal{O}} Q(t^+)$. If one of the chosen input queues of a coding link is or becomes empty while the other is not empty, coding produces degenerate poison packets. These are treated as normal poison packets by the back pressure except that branching transforms each such packet into a single individual poison packet which subsequently requests a dummy remedy packet for "decoding".
2) Remove all packets from queues $U_{d_c}^c, U_{d_c}^{cv}$.
3) Add r_c units to the source queues $U_{s_c}^c$.
4) After completing steps 1-3 at time $(t+T)^-$, for each session c, transfer packets between the source queue $U_{s_c}^c$ and the overflow queue $\bar{U}^c$ of each session c, so as to maximize $U_{s_c}^c(t^+)$ subject to a maximum length constraint of V.

Theorem 3: If input rates $(r_c+\epsilon)$ are achievable with pairwise XOR coding for some $\epsilon > 0$, then the online algorithm with $V = 3\mu^2\left(2.25N^2 + 21.25N + 6.25K\right)T/2\epsilon$ stabilizes the system for rates (r_c).

Proof: Define total potential $L(t) = \sum_{Q\in\mathcal{Q}} Q(t)^2 + 2\sum_c V\bar{U}_c(t)$, where $\mathcal{Q}$ is the set of all queues in the network apart from the overflow queues. Note that by arguments similar to those in [8], step 4 does not increase the total potential $L(t)$, so we focus on the change in potential across steps 1-3.

The queues evolve according to

$$Q(t+T) \le \max\{Q(t) - Ty_Q\ ,\ 0\} + Tx_Q$$

where

$$\begin{aligned}
y_{U_i^c} &= \sum_b (\nu_{ib}^c + \chi_{ib}^c) \\
x_{U_i^c} &= \sum_a \nu_{ai}^c + \eta_i^c + \begin{cases} r_c & i = s_c \\ 0 & i \neq s_c, d_c \end{cases} \\
y_{U_i^{cv}} &= \sum_b \nu_{ib}^{cv} + \sum_{c'\neq c,v'} \gamma_i^{\{cv,c'v'\}} \\
x_{U_i^{cv}} &= \sum_a \nu_{ai}^{cv} + \chi_{vi}^c \\
y_{P_i^{\{cv,c'v'\}}} &= \sum_b \pi_{ib}^{\{cv,c'v'\}} + \sigma_i^{\{cv,c'v'\}} \\
x_{P_i^{\{cv,c'v'\}}} &= \sum_a \pi_{ai}^{\{cv,c'v'\}} + \sum_{v,v'} \gamma_i^{\{cv,c'v'\}} \\
y_{P_i^{cv'}} &= \sum_b \pi_{ib}^{cv'} + \kappa_i^{cv'} \\
x_{P_i^{cv'}} &= \sum_a \pi_{ai}^{cv'} + \sigma_i^{\{cv,c'v'\}} \\
y_{R_i^{cj}} &= \sum_b \rho_{ib}^{cj} + \eta_{\{i=j\}}^c
\end{aligned}$$

[4]Note that the joint poison queues have one more superscript than in the off-line case, owing to the direction of poison flow.

$$x_{R_i^{cj}} = \sum_a \rho_{ai}^{cj} + \kappa_j^{ci}$$

and the time index (t) has been omitted from the flow variables for brevity.

Let x_Q and y_Q be the total allocated flow rate into and out of queue Q respectively, and $\mathcal{Q}_i$ the set of all queues at node i. Squaring and summing over all queues, and dropping some negative terms from the right hand side, we have

$$E\left\{L(t+T) - L(t)\right\} \le BT^2 - 2T \sum_{Q\in\mathcal{Q}} Q(t)\left(-x_Q + y_Q\right) \tag{16}$$

where

$$\begin{aligned} B &= \sum_{Q\in\mathcal{Q}} \left(x_Q^2 + y_Q^2\right) \\ &\le \sum_{i\in\mathcal{N}} \left(\sum_{Q\in\mathcal{Q}_i} y_Q\right)^2 + \sum_c \left(x_{U_{s_c}^c}\right)^2 + \\ &\quad \sum_{i,j} \left(\sum_c x_{R_i^{cj}}\right)^2 + \sum_{i\in\mathcal{N}} \left(\sum_{Q\in\mathcal{Q}_i, Q\ne U_{s_c}^c, Q\ne R_i^{cj}} x_Q\right)^2 \\ &\le N\left(7\mu/2\right)^2 + K\left(5\mu/2\right)^2 + N^2\left(3\mu/2\right)^2 + \left(3\mu\right)^2 \\ &= \mu^2\left(2.25N^2 + 21.25N + 6.25K\right) \end{aligned} \tag{17}$$

Let $\tilde{x}_Q(t-T)$ and $\tilde{y}_Q(t-T)$ be the average total flow rate into and out of queue Q respectively during the time slot $(t-T, t)$. We can rewrite $\sum_{Q\in\mathcal{Q}} Q(t)\left(-x_Q + y_Q\right)$ as

$$\begin{aligned} &\sum_{Q\in\mathcal{Q}} \left(Q(t-T) + \tilde{x}_Q(t-T)T - \tilde{y}_Q(t-T)T\right)\left(-x_Q + y_Q\right) \\ &\ge \sum_{Q\in\mathcal{Q}} Q(t-T)\left(-x_Q + y_Q\right) - BT \end{aligned} \tag{18}$$

Substituting into (16), we have

$$E\left\{L((t+T)^-) - L(t^-)\right\} \le 3BT^2 - 2TD \tag{19}$$

where

$$D = E\left\{\sum_{Q\in\mathcal{Q}} Q(t-T)\left(-x_Q + y_Q\right)\right\} \tag{20}$$

If input rates $(r_c + \epsilon)$ are feasible with pairwise XOR coding, there exists some value $\underline{\xi}$ of the vector of flow variables $(\nu_{ab}^c, \ldots)$ satisfying:

$$\begin{aligned} &\sum_{\{cv,c'v'\}} \gamma_i^{\{cv,c'v'\}} \le \mu/2, \sum_{\{cv,c'v'\}} \sigma_i^{\{cv,c'v'\}} \le \mu/2, \\ &\sum_{c,j} \kappa_i^{cj} \le \mu/2, \sum_c \eta_i^c \le \mu/2 \\ &\sum_b \left(\nu_{ib}^c + \chi_{ib}^c\right) = \sum_a \nu_{ai}^c + \eta_i^c + \begin{cases} r_c + \epsilon & i = s_c \\ 0 & i \ne s_c, d_c \end{cases} \\ &\sum_b \nu_{ib}^{cv} + \sum_{c'\ne c, v'} \gamma_i^{\{cv,c'v'\}} = \sum_a \nu_{ai}^{cv} + \chi_{vi}^c \\ &\sum_b \pi_{ib}^{\{cv,c'v'\}} + \sigma_i^{\{cv,c'v'\}} = \sum_a \pi_{ai}^{\{cv,c'v'\}} + \sum_{v,v'} \gamma_i^{\{cv,c'v'\}} \\ &\sum_b \pi_{ib}^{cv'} + \kappa_i^{cv'} = \sum_a \pi_{ai}^{cv'} + \sigma_i^{\{cv,c'v'\}} \\ &\sum_b \rho_{ib}^{cj} + \eta_{\{i=j\}}^c = \sum_a \rho_{ai}^{cj} + \sum_v \kappa_j^{ci} \\ &\sum_c \left(\sum_v \nu_{ab}^{cv} + \nu_{ab}^c\right) + \sum_{c,j} \left(\rho_{ab}^{cj} + \pi_{ab}^{cj}\right) \\ &+ \sum_{\{cv,c'v'\}} \pi_{ab}^{\{cv,c'v'\}} \le R_{ab} \end{aligned} \tag{21}$$

For a randomized policy which does power allocation and scheduling based on this solution vector $\underline{\xi}$, analogously to the algorithm of [11], we have, from (20),

$$D = \sum_c U_{s_c}^c (t-T)\epsilon \tag{22}$$

Next, we consider a slightly modified policy which does not send to any queue Q for which $Q(t-T) > 2V(L+1)$. This ensures that all queues remain shorter than $2V(L+1) + 5\mu$. As in the off-line case, we consider a flow solution which we partition into elementary flows $\mathcal{F}_n^c$ whose size is denoted f_n^c. A queue in $\mathcal{F}_n^c$ is considered upstream or downstream of another according to the direction of flow in the problem defined above. At each time slot $(t, t+T)$, we modify the solution vector $\underline{\xi}$ as follows. Initialize $\mathcal{H}$ as the set of all flows $\mathcal{F}_n^{*c}$, and $\mathcal{H}_1$ as the empty set. In the following, queue length refers to length at time $t-T$.

- Phase A: Repeat
  - Choose some flow $\mathcal{F}_n^c \in \mathcal{H}$ that contains some non-joint poison queue of length at least $2V(L+1)$ or whose primary path contains some uncoded queue of length at least $2V(L+1)$. Let $\bar{Q}_n^c$ be the furthest downstream such queue in $\mathcal{F}_n^c$. Remove from $\mathcal{H}$ all flows that share a path segment with $\mathcal{F}_n^c$. Remove $\mathcal{F}_n^c$ from $\mathcal{H}$, add it to $\mathcal{H}_1$, truncate the portion upstream of $\bar{Q}_n^c$ and convert the portion downstream of $\bar{Q}_n^c$ into an uncoded flow along the primary path.
  - If there is no such flow remaining, end Phase A.
- Phase B: Repeat
  - Choose some flow $\mathcal{F}_n^c \in \mathcal{H}$ that contains some joint poison queue of length at least $2V(L+1)$. Let $\bar{Q}_n^c$ be the furthest downstream such queue in $\mathcal{F}_n^c$, and $\mathcal{F}_{n'}^{c'}$ the other flow corresponding to $\bar{Q}_n^c$. Remove from $\mathcal{H}$ all flows that share a path segment with $\mathcal{F}_n^c$ or $\mathcal{F}_{n'}^{c'}$ downstream of $\bar{Q}_n^c$. Remove $\mathcal{F}_n^c$ and $\mathcal{F}_{n'}^{c'}$ from $\mathcal{H}$, add them to $\mathcal{H}_1$, truncate the portion of each flow upstream of $\bar{Q}_n^c$ and convert the portion of each flow downstream of $\bar{Q}_n^c$ into an uncoded flow along its primary path. Any path segment shared by a truncated portion and another flow becomes an uncoded segment for that other flow.
  - If there is no such flow remaining, end Phase B.
- For any flow not in $\mathcal{H} \cup \mathcal{H}_1$ or any truncated portion of a flow in $\mathcal{H}_1$, subtract the flow size from the corresponding entries of $\underline{\xi}$.

Note that the flow entering any queue of length $2V(L+1)$ or greater will be zero, and that this modification increases D by the difference between the sum of the lengths of the starting queues of the modified flows weighted by their flow size, and the sum of the source queue lengths weighted by the source rate, which is positive since each flow whose starting queue has length $2V(L+1)$ or greater corresponds to most $2(L+1)$ removed flows of the same size.

Next, we consider the back pressure algorithm. We can rewrite (20) as follows, omitting the time indexes $t-T$ from the queue lengths for brevity:

$$\begin{aligned} D = \sum_{ab} \Bigg(& \sum_{c} \nu_{ab}^{c} \left(U_a^c - U_b^c\right) + \sum_{c,v} \nu_{ab}^{cv} \left(U_a^{cv} - U_b^{cv}\right) \\ & + \sum_{c} \chi_{ab}^{c} \left(U_a^c - U_b^{ca}\right) \\ & + \sum_{\{c,c'\},j} \pi_{ab}^{\{cv,c'v'\}} \left(P_a^{\{cv,c'v'\}} - P_b^{\{cv,c'v'\}}\right) \\ & + \sum_{c,j} \left(\pi_{ab}^{cj} \left(P_a^{cj} - P_b^{cj}\right) + \rho_{ab}^{cj} \left(R_a^{cj} - R_b^{cj}\right)\right) \Bigg) \\ & + \sum_{\{cv,c'v'\},i} \gamma_i^{\{cv,c'v'\}} \left(U_i^{cv} + U_i^{c'v'} - P_i^{\{c,c'\}i}\right) \\ & + \sum_{\{cv,c'v'\}} \sigma_i^{\{cv,c'v'\}} \left(P_i^{\{cv,c'v'\}} - P_i^{cv'} - P_i^{c'v}\right) + \\ & \sum_{c,v',i} \kappa_i^{cv'} \left(P_i^{cv'} - R_{v'}^{ci}\right) + \sum_{c,c',i,j} \eta_i^{cv'} \left(R_i^{cj} - U_i^c\right) \end{aligned} \quad (23)$$

Since the back-pressure algorithm maximizes (23), we have, substituting into (16),

$$E\left\{L(t+T) - L(t)\right\} \leq 3BT^2 - 2T \sum_{c} U_{s_c}^{c}(t-T)\epsilon \quad (24)$$

If $\underline{U}^c(t) = 0$ for all c, then $L(t) \leq |\mathcal{Q}|(2V(L+1)+5\mu)^2$. If $\underline{U}^c(t) > 0$ for some c, then $U_{s_c}^c(t) = V$, and

$$E\left\{L(t+T) - L(t)\right\} \leq 3BT^2 - 2TV\epsilon.$$

Setting $V = \frac{3BT}{2\epsilon}$ gives $E\left\{L(t+T) - L(t)\right\} \leq 0$. By induction on the number of time slots, $E\{L(t)\} \leq |\mathcal{Q}|(2V(L+1)+5\mu)^2$ for all t and the queues are stable.

■

VII. Variations and extensions

A. Restricting the solution space

We can reduce the amount of computation required by avoiding direct optimization over all of the coding and decoding possibilities. Instead, we can discover useful possibilities by random exploration: for instance, in addition to opportunistic coding [9], congested nodes can do limited coding of pairs of packets that are not known a priori to be decodable by the next hop. Downstream nodes then report rates at which remedy packets of different sessions c originate from different nodes v. We can then create separate U^{cv} commodities based on the reported success rates. This results in optimization over a smaller set of commodities.

The algorithm also lends itself readily to other modifications that reduce complexity by restricting the space of solutions. For instance, we can limit the values of H and L. We can also restrict flows of a session from travelling too far away from the corresponding sink.

B. Allowing more sessions to be simultaneously coded

This approach can be extended to consider coding together more than two sessions, but at the expense of substantially higher complexity: more choices of which flows to code together, more queues to keep track of the different combinations of coded flows, and more remedy flows (which may be multicasts rather than unicasts). The complexity can be controlled by considering restricted classes of coding configurations, for instance, requiring remedies from a three-way coding operation to originate at a single location for each of the coded flows.

References

[1] Baruch Awerbuch and Tom Leighton. A simple local control approximation algorithm for multicommodity flow. In *Proceedings of 34th IEEE Conference on Foundations of Computer Science*, 1993.

[2] Baruch Awerbuch and Tom Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. In *Proceedings of 26th ACM Symposium on Theory of Computing*, 1994.

[3] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu. Joint asynchronous congestion control and distributed scheduling for wireless networks. In *Proc. IEEE Infocom*, 2006.

[4] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. *IEEE Transactions on Information Theory*, 51(8)::2745–2759, August 2005.

[5] R. Dougherty and K. Zeger. Nonreversibility and equivalent constructions of multiple unicast networks. *IEEE Transactions on Information Theory, submitted*, 2005.

[6] M. Effros, T. Ho, and S. Kim. A tiling approach to network code design for wireless networks. In *IEEE Information Theory Workshop*, 2006.

[7] A. Eryilmaz and D. S. Lun. Control for inter-session network coding, MIT LIDS Technical Report #2722, August 2006.

[8] T. Ho and H. Viswanathan. Dynamic algorithms for multicast with intra-session network coding. In *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[9] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Mdard. Practical network coding for wireless environments. In *Proceedings of 43rd Allerton Conference on Communication, Control, and Computing, Monticello, IL*, September 2005.

[10] Michael Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 11(1), 2000.

[11] Michael Neely, Eytan Modiano, and Charles E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. In *Proceedings of IEEE Infocom*, 2003.

[12] N. Ratnakar, R. Koetter, and T. Ho. Linear flow equations for network coding in the multiple unicast case. In *DIMACS workshop on network coding*, 2005.

[13] N. Ratnakar, D. Traskov, and R. Koetter. Approaches to network coding for multiple unicasts. In *Int. Zurich Seminar on Communications (IZS)*, February 2006.

[14] Leandros Tassiulas and Anthony F. Ephremides. Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop networks. *IEEE Transactions on Information Theory*, 1992.

[15] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard. Network coding for multiple unicasts: An approach based on linear optimization. In *Proceedings of the International Symposium on Information Theory*, 2006.

[16] L. Ying, R. Srikant, A. Eryilmaz, and G. E. Dullerud. Distributed fair resource allocation in cellular networks in the presence of heterogeneous delays. *IEEE Trans. Automatic Control, to appear*, 2005.