A Linear Inter-Session Network Coding Scheme for Multicast

Min Yang and Yuanyuan Yang
Dept. of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA

ABSTRACT

Network coding is a promising generalization of routing which allows a network node to generate output messages by encoding its received messages to reduce the bandwidth consumption in the network. An important application where network coding offers unique advantages is the multicast network where a source node generates messages and multiple receivers collect the messages. Previous network coding schemes primarily considered encoding the messages in a single multicast session. In this paper, we consider the linear inter-session network coding for multicast. The basic idea is to divide the sessions into different groups and construct a linear network coding scheme for each group. We use two metrics to guide the group division: overlap ratio and overlap width. These two metrics measure the benefit that a system can achieve by inter-session network coding with different considerations. The overlap ratio mainly characterizes the network bandwidth while the overlap width characterizes the system throughput. Our simulation results show that the proposed inter-session network coding scheme can achieve about 30% higher throughput than intra-session network coding.

Keywords: Network coding, Linear coding, Inter-session coding, Multicast network.

I. INTRODUCTION

Today's network transmits messages usually by routing, that is, by having intermediate nodes store and forward messages. However, routing does not encompass all operations that can be performed at a node. Recently, the notion of *network coding* arises as a promising generalization of routing. Network coding refers to a scheme where a node is allowed to generate output messages by encoding (i.e., computing certain functions of) its received messages. Thus, network coding allows information to be mixed, in contrast to the traditional routing approach where each node simply forwards received messages. Network coding was first brought forward to achieve the multicast capacity in a multicast network in [1]. Given a multicast network represented by a directed graph G, the multicast capacity is defined as the minimum of the maximum flows between the source and each receiver. It is considered as the upper bound of the throughput a multicast network can achieve. The advantage of network coding on a multicast network can be demonstrated by the well-known butterfly network as shown in Figure 1(a), where node s is the source, and nodes r_1 and r_2 are two receivers. All edges in the figure have capacity 1, which means that each edge can transmit only 1 unit data (bit) per unit time (second). Source s has two bits, b_1 and b_2, to multicast to both r_1 and r_2. First, we use the traditional multicast without network coding as shown in Figure 1(b). Without loss of generality, we use the dashed line (red) to represent bit b_1, the dotted line (blue) to represent bit b_2 and the bold line (green) to represent both bits b_1 and b_2. Bit b_1 can reach r_1 in two seconds. Bit b_2 can also reach r_2 in two seconds. When node c receives both bits, it forwards them sequentially. Suppose it forwards bit b_1 first. Then r_1 receives both bits in 4 seconds and r_2 receives both bits in 5 seconds. Now consider using network coding on link $c-d$ as shown in Figure 1(c). When node c receives both bits, it first mixes them by performing an exclusive OR (XOR) operation on the bits. Then it sends the mixed bit b to node d. When nodes r_1 or r_2 receive the mixed bit, they can recover the original bits b_1 and b_2 by XORing the mixed bit and the other received bit. This way, all the transmissions can be completed in 4 seconds.

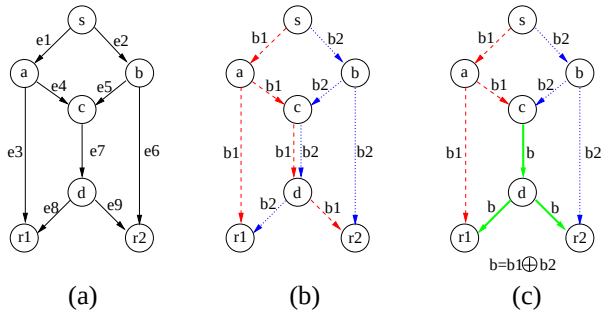

Fig. 1. (a) Butterfly network. (b) Multicast without network coding. (c) Multicast with network coding.

A multicast network can be modeled by a directed graph where each node has multiple incoming edges and outgoing edges. Constructing a network coding scheme for a multicast network is equivalent to assigning a function to each edge which defines the mix operation for the edge. This function is called *edge function*. Suppose the edge is one of the outgoing edges of node v. Then its edge function takes the messages on the incoming edges of node v as input and outputs a message which is

Manuscript received April 30, 2009; revised June 60, 2009; accepted July 30, 2009. Research was supported in part by National Science Foundation under grant numbers CCR-0207999 and CCF-0744234.

doi:10.4304/jcm.4.11.865-872

transmitted through itself.

Linear network coding refers to the network coding scheme in which the edge functions are constrained to linear functions over a finite field. In linear network coding, the edge function is also called edge vector because it is represented by a vector whose coordinates are the coefficients of the linear edge function. A linear network coding scheme is valid if the receivers can recover the original messages from the received encoded messages.

Most existing works on network coding in the literature focused on a single multicast session. For example, Li et al. [2] showed that linear network codes are sufficient to achieve the multicast capacity. Kotter et al. gave an algebraic characterization for a linear network coding scheme in [3]. They also gave an upper bound on the field size and a polynomial time algorithm to verify the validity of a network coding scheme. Ho et al. presented a random linear network coding approach in [4], [5] in which nodes generate edge vectors randomly. The linear network coding scheme generated by this approach is not always valid. They proved that the probability of failure is $O(1/q)$ where q is the size of the finite field. In contrast to the random network coding, Jaggi et al. proposed a polynomial deterministic algorithm in [8] which can construct deterministic linear network coding schemes for multicast networks.

An extension to the network coding for a single multicast session is to apply network coding to multiple concurrent multicast sessions, which is called *inter-session network coding*. The benefit of inter-session network coding can be demonstrated by the example shown in Figure 2, where nodes s_1 and s_2 are the respective sources of the two multicast sessions and nodes r_1 and r_2 are the receivers of both multicast sessions. We can see that edge $c-d$ becomes a bottleneck if no inter-session network coding is employed. Inter-session network coding can eliminate the bottleneck by encoding two messages received at node c and send them along link $c-d$ together. Both r_1 and r_2 can recover the messages from s_1 and s_2.

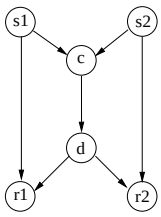

Fig. 2. An example that inter-session network coding achieves higher throughput.

However, although it is an extension of the single multicast session, inter-session network coding is much more complex. Dougherty et al. [10] showed that linear network coding is insufficient to achieve multicast capacity for multiple multicast sessions. Li et al. [7] showed that there is no coding gain for an undirected graph. Also, even we confine the edge function to linear functions, it is a NP-hard problem [3] to find such a linear network coding scheme. Wang et al. [6] gave some preliminary work on inter-session network coding for two simple multicast sessions from a graph theory point of view. They proved an equivalent condition under which there exists a linear network coding scheme for two multicast sessions. Wu [11] applied random network coding to all the sessions after transforming the network topology such that the source can only reach the receivers that are interested in the source.

To the best of our knowledge, this is the first work that provides a practical method to construct a linear network coding scheme for inter-session network coding and evaluate its performance. In this paper, we will investigate inter-session network coding by providing heuristic algorithms and conducting extensive simulations. We will propose an approach to identifying the situations where it is the most profitable to do inter-session network coding and which sessions should be encoded together. Two metrics will be introduced to characterize the overlap among sessions. The sessions are divided into multiple groups based on the metrics such that the overlap among sessions in the same group is above a threshold. The inter-session network coding is constrained within the same group. We will also propose two heuristic algorithms, the deterministic algorithm and the random algorithm, to construct the linear coding scheme on the divided groups. Our simulation results show that the system with inter-session network coding achieves much higher throughput than that without inter-session network coding.

The rest of the paper is organized as follows. In Section II, we give a formal description of the inter-session network coding problem. In Section III, we present our heuristic solutions from both deterministic and random perspectives. We conduct extensive simulations and discuss the simulation results in Section IV. Finally, Section V concludes the paper.

II. PRELIMINARIES

We model the network as a directed acyclic graph (DAG), $G = (V, E)$, where V is the node set and E is the edge(link) set. An edge e can be represented by an ordered node pair (x, y) where $x, y \in V$. y is called the head of the edge and x is called the tail of the edge. The messages can only be transmitted from x to y.

The incoming edge set of a node v is defined as

$$E_{in}(v) = \{(x, y) | (x, y) \in E, y = v\} \quad (1)$$

Similarly, the outgoing edge set of a node v is

$$E_{out}(v) = \{(x, y) | (x, y) \in E, x = v\} \quad (2)$$

Each node has one or more incoming edges and one or more outgoing edges except that source nodes have no incoming edges and receivers have no outgoing edges. Each edge, also called link, has a link capacity of 1, which means that it can only transfer 1 unit of data at 1 time slot. For a network with link capacities larger than 1, we transform the network based on the following rule: for each link with link capacity lc ($lc > 1$), we replace the link with lc links such that each of them has the same head

and tail as the original link and has a link capacity of 1. In the case that a receiver has one or more outdoing edges, we add a virtual receiver to replace it and multiple virtual links from the original receiver to the virtual receiver. The number of the virtual links equals to the number of incoming edges of the original receiver.

A multicast session is represented by a pair (s, M) (we assume $s \notin M$) where $s \in V$ represents the source of the session, and $M \subset V$ represents the set of receivers. We assume that there are k concurrent multicast sessions represented by (s_1, M_1), (s_2, M_2), ..., (s_k, M_k). For multicast session (s_i, M_i), the multicast capacity is denoted by C_i, and the transmission rate is denoted by r_i. Clearly, $r_i \leq C_i$. We call vector $\pi = (r_1, r_2, \ldots, r_k)$ a *rate vector*. A rate vector is achievable if it is possible to transmit all the sessions at the respective rates in the rate vector. The *achievable rate region*, or *rate region* for short, is the set of all achievable rate vectors.

The inter-session network coding problem for multiple multicast sessions can be described as follow:

Given a DAG, $G = (V, E)$ and k multicast sessions, (s_1, M_1), (s_2, M_2), ..., (s_k, M_k), find the rate region for the k multicast sessions and a method to achieve the rate region.

A special case of inter-session network coding is that each multicast session constructs a network coding scheme for its own session. As the network coding is only within the same session, we call it intra-session network coding. Intra-session network coding is easy to implement. However, it is not the optimal solution in most cases, because it can achieve the optimal rate region only if we can find a subgraph for each session such that these subgraphs are edge-disjoint and the multicast capacity of the subgraph is no less than the multicast capacity of the corresponding session. Apparently, this is difficult to achieve in general. In fact, the butterfly network in Figure 1 is an example that intra-session network coding is inferior to inter-session network coding.

In this paper, we are interested in solving the inter-session network coding problem through linear network coding due to its simplicity and easy to implement in hardware. As discussed earlier, linear network coding alone can not achieve the optimal rate region. Thus, we focus on maximizing the rate region with linear inter-session network coding. In particular, we are interested in answering following questions:

- *Question 1*: Under what condition inter-session network coding outperforms intra-session network coding? We are especially interested in throughput improvement. As intra-session network coding is a special case of inter-session network coding, the maximum rate region of inter-session network coding is no less than that of intra-session. Therefore, the maximum throughput of inter-session network coding is no less than that of intra-session.
- *Question 2*: How much can inter-session network coding improve the performance compared to intra-session network coding? We are interested in quantifying the benefit brought by applying the inter-session network coding. We believe the benefit is a function of some factors which leads to the next question.
- *Question 3*: What are the main factors that affect the performance? We are interested in finding the factors which dominate the benefit of inter-session network coding. These factors are then taken into consideration when constructing a practical inter-session network coding scheme.

III. Heuristic Algorithms for Linear Inter-Session Coding for Multicast

In this section, we propose heuristic algorithms for constructing a linear network coding scheme to achieve a near optimal rate region. To fully examine the behaviors and properties of inter-session network coding, we propose two heuristic algorithms: one is deterministic and the other is random.

A naive way to apply inter-session network coding to multiple sessions is to combine all the sessions into one "big" session. The source node of the big session is an artificial node connecting to the original source nodes of the multiple sessions, and the receivers of the big session are the union of the receivers of the multiple sessions. Then an intra-session linear network coding is applied to this big session. However, this naive solution can hardly improve the performance, or even worsen the performance. This is because different sessions have messages destined to different sets of receivers. When considered as one big session, the probability a receiver receives a message it is not interested increases, which causes more wasting of bandwidth. To avoid such situation, we divide the sessions into groups and perform intra-session network coding within each group. The selection of group is performed carefully such that the benefit of inter-session network coding overwhelms the overhead. We adopt two metrics for the group division. We first describe the two metrics.

A. Two metrics for session division

The goal of mixing different sessions is to increase the throughput by eliminating the bottlenecks caused by shared links of different sessions. If there are no shared links between two sessions, inter-session network coding is not necessary or even impossible. Therefore, we have the following heuristic rule for the algorithm: sessions that overlap more will benefit more from inter-session network coding.

Now the problem becomes how to characterize the overlap among sessions. We expect to find a method to quantify the shared links among sessions in order to determine which group of sessions should be considered together. Before we dive into the details, we introduce a notion called $field$ to facilitate our presentation. A $field$ is a function which maps a session to a subgraph of the network topology. Recall that given a multicast session (s_i, M_i), it is always possible to find C_i edge-disjoint paths from the source to any of the receivers. Thus, there are a

total of $C_i|M_i|$ such paths. A $field$ is formally defined as follows:

$$\begin{aligned} &field(s_i) = G'(V', E') \text{ where} \\ &V' = \{v'|v' \in \text{one of the } C_i|M_i| \text{ paths}\}, \\ &E' = \{e|e \in E, head(e) \in V' \text{ and } tail(e) \in V'\} \end{aligned} \quad (3)$$

Each session has its own field. It is one of the subgraphs over which the session can achieve the multicast capacity through network coding. Fields may overlap, that is, a link may belong to multiple fields. Now we can quantify the overlap among sessions by adopting the following two metrics:

- *Overlap Ratio (OR)*: the overlap ratio measures the overlap by the percentage of the overlapped links between two sessions. Suppose the two sessions are s_i and s_j. The overlap ratio of the two sessions can be calculated by the following function:

$$OR(s_i, s_j) = \frac{CL(s_i, s_j)}{|E'(s_i)| + |E'(s_j)| - CL(s_i, s_j)} \quad (4)$$

  where $CL(s_i, s_j)$ represents the number of common links of $field(s_i)$ and $field(s_j)$, and $|E'(s)|$ represents the number of links in $field(s)$. From the definition, we can see that this metric gives a higher priority to the sessions that have the most common links.
- *Overlap Width (OW)*: the overlap width measures the overlap by the percentage of the overlapped paths between two sessions. Here the paths refer to the edge-disjoint paths in the field. The overlap width of the two sessions can be calculated by the following function:

$$OW(s_i, s_j) = \frac{\sum_{m=1}^{C_i} \sum_{n=1}^{C_j} P_{mn}}{C_i C_j} \quad (5)$$

  where $P_{mn} = 1$ if the m^{th} path in $field(s_i)$ shares one or more links with the n^{th} path in $field(s_j)$ or $P_{mn} = 0$ otherwise. From the definition, we can see that this metric gives a higher priority to the sessions that have the most number of paths crossed.

The two metrics have their respective considerations. OR focuses on the overlapped links while OW focuses on the overlapped paths. A third metric is to consider both overlapped links and paths at the same time. As it simply combines the two proposed metrics, we do not consider it in this paper.

We use a tunable parameter δ ($0 \leq \delta \leq 1$) as a threshold. If either $OR(s_i, s_j)$ or $OW(s_i, s_j)$ is greater than δ, we put session (s_i, M_i) and (s_j, M_j) into the same group. After checking all the sessions, we can divide the multiple sessions into several groups with each group composed of one or more sessions. The groups are disjoint with each other. Now we can apply network coding to each group respectively.

B. The deterministic algorithm

The deterministic algorithm constructs the linear network coding scheme by assigning a fixed edge vector to each edge. The edge vectors are designed such that the receiver can recover the original messages based on its received messages. Without inter-session network coding, nodes can only mix the messages generated by the same source. If inter-session network coding is permitted, nodes can mix the messages from different sources. We introduce a parameter called $mixability$ to describe the maximum number of sessions that are permitted to be mixed together. If $mixability = 1$, inter-session network coding degenerates to intra-session network coding. If $mixability = k$, all the sessions are considered as one unified session and the network coding scheme is constructed on the unified session. We will evaluate the effect of $mixability$ on the performance in Section IV.

As discussed earlier, the multicast sessions are divided into different groups such that groups are disjoint with each other. All the sessions within the same group are considered as one unified session. The union of the sources forms the sources of the unified session. The union of the receiver sets of the sessions forms the receiver set of the unified session. We add one artificial source node connected to all the sources to simplify the linear network coding construction. Now the inter-session network coding for multiple sessions is transformed to an intra-session network coding for the unified session. There are several existing methods to construct a deterministic linear network coding scheme for a single multicast session. Here we adopt the method proposed in [13] for each group of sessions respectively. Given a group, we first preprocess the graph to find a minimum subgraph which has the same multicast capacity as the original graph. This preprocessing can greatly reduce the graph size to be processed. The subgraph can be looked as the union of the fields of the sessions in the group. Usually a field of a session is a subgraph which is much smaller than the whole graph. Then we can apply the hypergraph based approach in [13] to the subgraph to find a valid linear network coding scheme.

C. The random algorithm

The linear network coding scheme can be constructed not only in a deterministic way, but also in a random way. The fundamental difference between the two ways is how the coordinates of the edge vectors are generated. In random linear network coding, nodes mix the received messages and assign random coordinates to the edge vectors. Receivers keep receiving the encoded messages until they have enough independent messages to decode. In a random linear network coding without inter-session network coding, nodes can only mix messages generated by the same source. With inter-session network coding, nodes can mix the messages from different sources.

If there is no constraint for a node to mix and forward messages, that is, nodes always encode all the messages they receive and send to all the outgoing edges, the receivers will eventually be able to decode the original messages, since the sources will keep sending the messages until all the receivers decode the messages successfully. However, this method is inefficient as it

mixes all the sessions and treats them as one session. To avoid this problem, we divide the sessions into groups in a similar way to that used in the deterministic algorithm. Only messages within the same group can be encoded together, and nodes should obey the following rules:

1) If a node receives a message which contains the information generated by a source that does not belong to the group and the sessions whose fields include the node, the message should be discarded;
2) Encode the received message with other messages in the same group and send to all the outgoing edges.

Based on the above two rules, we constrain the messages generated from one source within its corresponding group. The reason is two folds. First, the receivers whose corresponding session is outside the group will never receive messages from the group. Thus the receivers can collect a sufficient number of independent messages to decode in a shorter time. Second, the messages will not flow aimlessly and the network bandwidth is saved.

IV. Performance Evaluations

We have conducted extensive simulations to evaluate the performance of the proposed algorithms when the inter-session network coding is employed. In this section, we present our simulation results and compare different approaches.

A. Simulation setup

We use NS-2 [16] as the network simulator. The network topologies are generated by GT-ITM software [17] which is a degree-based Internet structural topology generator. Each topology consists of 1000 nodes with an average node degree of 2 to 10 depending on the simulation scenario. Both the source nodes and the receiver nodes are selected randomly. Any node can not be a source node and a receiver node at the same time. To eliminate the randomness caused by the random topologies, for each simulation, we generate 10 random topologies. We run the simulation on the 10 random topologies and take the average of the 10 simulation results as the final result.

The simulation includes two parts. First, we implement the intra-session network coding and use it as a benchmark. We compare the two heuristic algorithms with the benchmark under different situations. Second, we study the behaviors of the inter-session network coding by tuning the parameters: $mixability$ and δ.

The simulation adopts the following two performance metrics:

- *Throughput*: Throughput is defined as the service the system provides in one time unit. Each node maintains an incoming buffer for each incoming edge and an outgoing buffer for each outgoing edge. The size of the buffer is 1. All the source nodes keep sending messages to the next node as long as the corresponding incoming buffer is not full. Each message contains a sequence number which indicates its position in the message stream. After a certain period of time, we stop all the message streams. The number of delivered messages is represented by the largest sequence number of the message that is received by all the receivers. The throughput of the system is the average of these numbers.
- *Bandwidth consumption*: We defined the network bandwidth used to deliver the messages as bandwidth consumption. As there are no control messages involved in the delivery, the bandwidth consumption is only caused by data messages. A data message going through one link contributes 1 unit to the bandwidth consumption.

B. Performance of inter-session network coding

We first compare the heuristic algorithms with the intra-session network coding in some general scenarios.

1) Multicast capacity: Figure 3(a) shows the throughput evaluation under different multicast capacities (average multicast capacity of all sessions). There are five curves which represent the deterministic algorithm using OR metric (denoted by "Determ, OR"), deterministic algorithm using OW metric (denoted by "Determ, OW"), random algorithm using OR metric (denoted by "Random, OR"), random algorithm using OW metric (denoted by "Random, OW"), and intra-session network coding (denoted by "Intra-session"), respectively. From the figure, we can see that both the deterministic algorithm and the random algorithm achieve higher throughput than the intra-session coding. It indicates that inter-session network coding can achieve better throughput than intra-session network coding. If we adopt OW metric, the throughput of the deterministic algorithm is about 30% higher than that of the intra-session coding. Also, the throughput of the deterministic algorithm is higher than that of the random algorithm. This is due to the possibility of failing to decode with random coding. We can see that with the increase of the multicast capacity, the throughput increases as well. The algorithm with metric OW performs better than that with metric OR with respect to the throughput. This can be explained from the definitions of the two metrics. OR emphasizes the shared links between sessions, in which the sessions with most common links are encoded together. OW emphasizes the crossed paths, in which the sessions with most crossed paths are encoded together. The throughput is determined by the bottleneck of the system. Encoding the sessions with most crossed paths together implies that the bottleneck is relaxed to the maximum extent. The gap between the heuristic algorithms and the intra-session coding becomes larger with the increase of the multicast capacity. It indicates that the system benefits more from the inter-session coding when the multicast capacity is larger due to the higher possibility to encode different sessions together.

Figure 3(b) shows the bandwidth consumption evaluation under different multicast capacities. As the figure shows, the bandwidth consumption increases with the increase of the multicast capacity. The deterministic algorithm achieves lower bandwidth consumption than

the random algorithm. We observe that the bandwidth consumption of the deterministic algorithm is slightly higher than that of the intra-session coding. This can be explained by the mechanism of inter-session network coding. For the sessions in the same group, the messages are encoded and transmitted together on bottleneck links which are shared by sessions. To this end, the traffic is reduced. On the other hand, for receivers to recover the original messages, it is inevitable to generate more messages for the receivers to decode the messages, which increases the traffic. However, these messages are usually transmitted through some relatively lightly loaded links. Based on the simulation, the increased traffic is slightly higher than the saved traffic. Given that inter-session network coding can increase the throughput by about 30% compared to the intra-session network coding, the minor increase in bandwidth consumption on some lightly loaded links is quite acceptable. We can also see that the algorithm with metric OR saves more network bandwidth than the algorithm with the metric OW. Since OR maximizes the shared links between the sessions encoded together, data packets are delivered only once on the shared links, which reduces the bandwidth usage. It indicates that metric OR is more suitable for applications where network bandwidth is scarce and expensive.

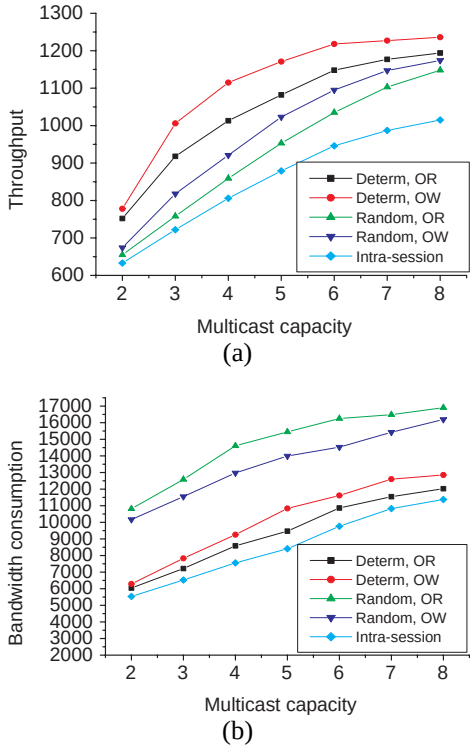

Fig. 3. Performance comparison under different multicast capacities. (a) throughput; (b) bandwidth consumption.

From the above discussion, we can see that if we adopt metric OR instead of metric OW, the bandwidth consumption can be saved. On the other hand, adopting OW can achieve higher throughput than OR. This is a general observation throughout the entire simulation. Since throughput is generally a critical performance metric in most networks and the difference of bandwidth consumption between metric OR and metric OW is small, in the rest of the simulation figures, we will draw the curve of metric OW only for clarity. Thus, by the deterministic (random) algorithm, we refer to the deterministic (random) algorithm adopting metric OW.

2) Session size: Figure 4 shows the throughput and bandwidth consumption under different session sizes (the session size is the number of receivers in the session). We let all the sessions have the same group size. As the figure shows, with the increase of the session size, the throughput drops sharply. This is because that a larger session size increases the possibility of overlap between sessions. The interference caused by overlap will drag the throughput down. However, the drop rate of the heuristic algorithms is lower than that of the intra-session due to the inter-session network coding.

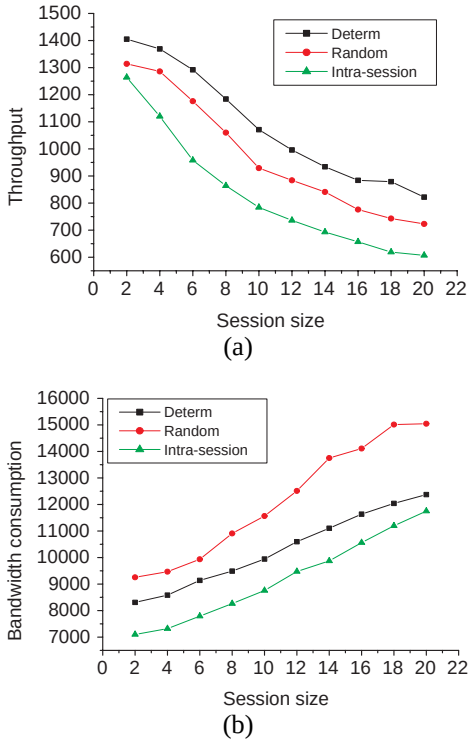

Fig. 4. Performance comparison under different session sizes. (a) Throughput; (b) Bandwidth consumption.

We can also see that the bandwidth consumption increases when the session size increases. The bandwidth consumption of the random algorithm is highest and it increases dramatically when the session size is greater than 8. This is due to the increased possibility of failing to decode the messages at the receivers.

From the above simulation results, we can draw the following conclusions: the deterministic algorithm achieves higher throughput than the random algorithm and the intra-session network coding (specifically, about 13% higher than the random algorithm and 30% higher than the intra-session network coding). The random algorithm has

the highest bandwidth consumption (specifically, about 35% higher than the deterministic algorithm and 45% higher than the intra-session network coding).

C. Inter-session network coding parameters

There are two important tunable parameters in the simulation of the inter-session network coding: $mixability$ and the threshold δ. $Mixability$ is used to limit the maximum number of sessions which are encoded together. When $mixability = 1$, the system degenerates to the intra-session network coding. The threshold δ is used to control the condition under which the sessions can be encoded together. When the threshold is high, the possibility to encode different sessions is low. In this subsection, we will examine how these two parameters affect the system performance through simulations.

1) Mixability: Figure 5(a) shows the evaluation for the throughput and bandwidth consumption under different $mixability$ values. We can see that the throughput of both the deterministic algorithm and the random algorithm experiences a rise followed by a drop. During the rise period, the deterministic algorithm has a steeper slope which indicates that the deterministic algorithm exploits the inter-session coding better than the random algorithm when $mixability$ is small. However, when $mixability$ is greater than 6, the throughput of the deterministic algorithm drops dramatically with the increase of $mixability$. The throughput of the random algorithm achieves its maximum when $mixability$ is around 9 after which it drops slowly. When the $mixability$ is large, the throughput of the random algorithm is higher than that of the deterministic algorithm.

At first glance, this performance degradation of heuristic algorithms may be surprising, since one may expect that the throughput of the inter-session network coding should not lower than that of the intra-session network coding. However, this is due to the fact that when two sessions with a large difference in their multicast capacities are encoded together, the average throughput of these two sessions is lower than that when only the intra-session network coding is used. This is because that if the two sessions have a large difference in multicast capacities, the session with a greater multicast capacity is dragged down by the other session as the capacity of the shared link is divided into half in inter-session coding instead of being allocated according to the different rate requirements by sources.

Based on this observation, we revise the heuristic algorithms by limiting the difference between the greatest multicast capacity and the smallest one in the same group. To do this, we compare their multicast capacities before we calculate the OR or OW metric for two sessions. If the difference is large than a specific value (denoted by ρ), no overlap metric is calculated, and these two sessions will not be put into one group. ρ is an upper bound for the difference of the multicast capacities between two sessions in one group. Otherwise, whether the two sessions are put into one group is based on the overlap metric. The determination of the optimal value of ρ depends on the average session size of the two sessions. The larger the session size, the smaller the optimal value of ρ. It indicates that if two sessions have a large average session size, it is more critical to have a small multicast capacity difference. The reason is that the throughput degradation due to the multicast capacity difference is more severe in this case as it involves more receivers when the session size is large.

Figure 5(b) shows the simulation results based on the revised algorithms. As an example, we plot the figure for the case when the average session size is 16 and $\rho = 4$. We can see that now the throughput always increases as $mixability$ increases although the increase rate is slower than the previous simulation. It indicates that the improved algorithm can eliminate the throughput degradation due to the large multicast capacity difference.

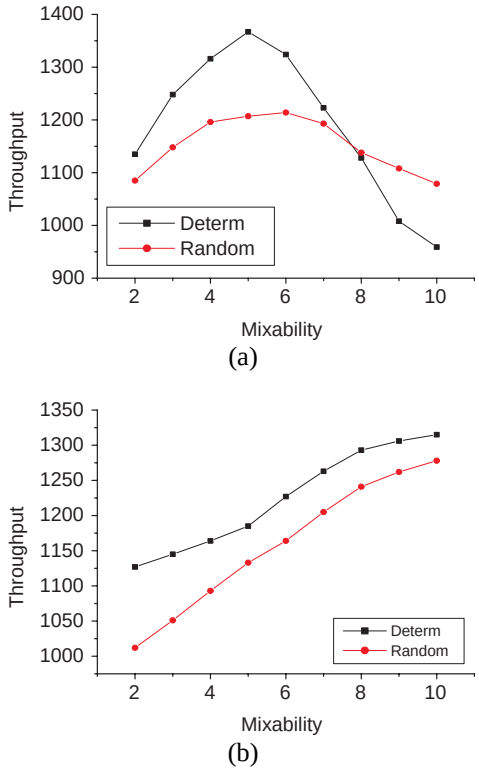

Fig. 5. Performance comparison under different $mixability$ values. (a) Throughput without considering the multicast capacity difference; (b) Throughput considering the multicast capacity difference.

Figure 6 shows the bandwidth consumption under different mixability values. With the increase of the mixability, the bandwidth consumption increases slightly. When the mixability is large, bandwidth consumption stays at the same level. This indicates that when $mixability$ is greater than 8, the group division remains the same.

2) Threshold δ: Figure 7(a) shows the throughput when we tune the threshold δ from 0.1 to 0.9. As can be seen, the throughput drops as the threshold increases. The throughput drops faster when δ is greater than 0.4. This is because that when δ is large, the possibility that overlapped sessions are encoded together becomes smaller. When δ is small, the throughput drops slightly,

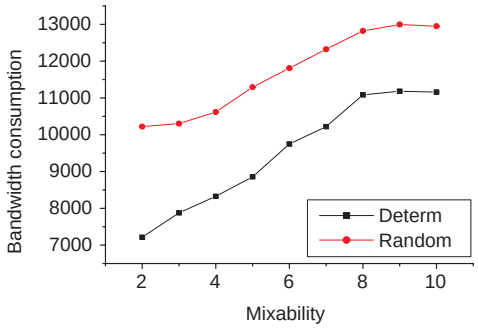

Fig. 6. Bandwidth consumption comparison under different $mixability$ values.

which indicates that the number of sessions affected by the threshold is small. It implies that for most of the sessions, the value of overlap metric is greater than 0.4.

Figure 7(b) shows the bandwidth consumption under different threshold values. The bandwidth consumption decreases with the increase of δ. This is another evidence that the traffic becomes less when the inter-session coding possibility is smaller.

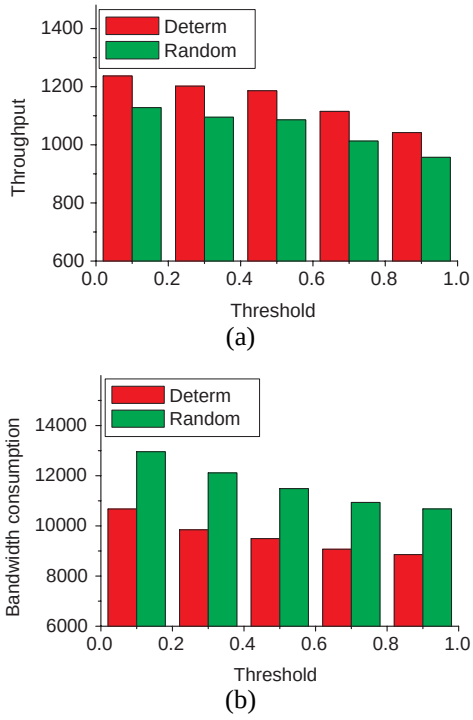

Fig. 7. Performance comparison under different δ values. (a) Throughput; (b) Bandwidth consumption.

From the above simulation results, we can draw the following conclusions: With the increase of $mixability$, both the throughput and the bandwidth consumption become higher; With the increase of δ, both the throughput and the bandwidth consumption become lower. When designing an inter-session network coding scheme, it is necessary to consider all the influential parameters: $mixability$, δ and ρ.

V. Conclusions

Network coding is a promising technique to improve the resource efficiency for multicast networks. In this paper, we have investigated the linear inter-session network coding for multicast. The contribution of this paper is three folds. First, we proposed a practical inter-session network coding scheme for multicast and implemented in NS-2. Second, we introduced two different metrics to characterize the benefit of inter-session network coding with each metric having its own application targets. Third, we studied the performance of inter-session network coding from both the deterministic coding and random coding perspectives. Our simulation results show that the inter-session network coding outperforms the intra-session network coding by about 30% in terms of throughput in most cases. In addition, the deterministic algorithm achieves higher throughput and less bandwidth consumption than the random algorithm. Our future work includes designing an inter-session network coding scheme which can handle network dynamics.

References

[1] R. Ahlswede, N. Cai, S.Y.R. Li and R.W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. 46, July 2000, pp. 1204-1216.
[2] S.Y.R. Li, R.W. Yeung and N. Cai, "Linear network coding," *IEEE Trans. Information Theory*, vol. 49, Feb. 2003, pp. 371-381.
[3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, 2003, pp. 782-795.
[4] T. Ho, M. Medard, J. Shi, M. Effros and D.R. Karger, "On randomized network coding," *Proc. Annual Allerton Conference on Communication, Control and Computing*, Sept. 2003.
[5] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Information Theory*, vol. 52, 2006, pp. 4413-4430.
[6] C. Wang and N.B. Shroff, "Intersession network coding for two simple multicast sessions," *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2007.
[7] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2004.
[8] N. Harvey, D.R. Karger and K. Murota, "Deterministic network coding by matrix completion," *ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, Canada, Jan. 2005.
[9] P. Sanders, S. Egner and L. Tolhuizen, "Polynomial time algorithms for network information flow," *Proc. 15th ACM Symp. Parallel Algorithms and Architectures*, San Diego, CA, June 2003.
[10] R. Dougherty, C. Freiling and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. Information Theory*, vol. 51, August 2005, pp. 2745-2759.
[11] Y. Wu, "On constructive multi-source network coding," *Proc. IEEE ISIT*, Seattle, WA, July 2006.
[12] A. Eryilmaz and D.S. Lun, "Control for Inter-session Network Coding," *Proc. of NetCod 2007*.
[13] M. Yang and Y. Yang, "A hypergraph approach to linear network coding in multicast networks," to appear in *IEEE Trans. Parallel and Distributed Systems*, 2009.
[14] Al-Bashabsheh. Ali and Yongacoglu. Abbas, "Average throughput with linear network coding over finite fields: the combination network case," *EURASIP J. Wireless Communications and Networking*, no. 4, January 2008.
[15] T. Ho, Y.-H. Chang and K. Han, "On constructive network coding for multiple unicasts," *Proc. Annual Allerton Conference on Communication, Control and Computing*, Sept. 2006.
[16] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/.
[17] GT-ITM: Georgia Tech Internetwork Topology Models, http://www.cc.gatech.edu/projects/gtitm/.