A Systematic Approach to Network Coding Problems using Conflict Graphs

Jay Kumar Sundararajan
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Email: jaykumar@mit.edu

Muriel Médard
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Email: medard@mit.edu

Ralf Koetter
Coordinated Science Laboratory
University of Illinois Urbana-Champaign
Urbana, IL 61801, USA
Email: koetter@uiuc.edu

Elona Erez
Dept. of Electrical Engineering-Systems
Tel Aviv University,
Tel Aviv, 69978, Israel
Email: elona@eng.tau.ac.il

***Abstract*— We present a new approach to network coding problems that could lead to a systematic method for deciding solvability of a given network. The approach is based on a graph theoretic formulation of the problem. The constraints at each node in the network are represented using hyperedges in a 'conflict' hypergraph. This representation reduces the solvability question to that of finding a stable set with certain properties in a hypergraph. The approach is sufficiently general to allow even non-linear codes by suitably modifying the conflict graph. We also demonstrate the use of the conflict graph idea in the context of a multicast crossbar switch. Using examples, we show that the rate region of a multicast switch strictly improves with network coding at the inputs, and derive an outer bound on the rate region when intra-session network coding is allowed.**

I. Introduction

Consider a general communication network consisting of error-free links, each with a specified capacity with some nodes being information sources and some being information sinks. In a fluid or transportation network, each commodity entering a node may only be routed onto some outgoing link. However, in information flow networks, information entering a node may also be replicated or even encoded before being sent out. In fact, Ahlswede *et al.* [1] showed that such "network coding" can achieve a strictly larger rate region as compared to routing, for certain networks. The question then is to characterize the rate region for network information flow, and to give algorithms to construct network codes that satisfy the rate demands of all source-sink connections in a network.

A lot of work has been done to answer these questions in the special case of a multicast connection, where every sink demands all information from every source. The maximum achievable rate was characterized in terms of the min-cut max-flow bound by [1] and [2]. Cai *et al.* [3] showed that linear codes suffice to achieve the min-cut bound, while Jaggi *et al.* [4] gave a polynomial time algorithm to construct linear codes for multicast. A randomized distributed approach was proposed by Ho *et al.* in [8] while deterministic coding strategies were proposed by Fragouli *et al.* [9] and by Harvey *et al.* [10].

For the more general case of non-multicast connections, although several results are known, the problem predominantly remains open. Lehman *et al.* showed that scalar linear codes do not suffice for the non-multicast case. Médard *et al.* gave an example network which requires linear coding involving vectors over a finite field. Zeger *et al.* gave an example network that cannot be solved using a linear code over any finite field for any vector dimension but can be solved using a non-linear code. Reference [2] gave an algebraic formulation of the general network coding problem and showed that deciding solvability is equivalent to deciding whether a certain algebraic variety is empty or not.

Our paper aims to provide a new formulation of the general network coding problem using a graph theoretic model. The basic idea is to note that a valid network code is essentially an assignment of states to links such that several local constraints are met. If the possible states of each link are represented using vertices, and the local constraints are represented using edges (or hyperedges) among them, then the problem of finding a network code is essentially that of finding a stable set[1] in the "conflict hypergraph". This formulation could potentially lead to a systematic way of deciding the solvability of a network coding problem. The main strength of this formulation is that it can easily accommodate conditions such as linearity of the code, or coding only at specific nodes, by suitably modifying the conflict graph.

The idea of using a conflict graph to represent a problem with several local constraints has been used before in the context of a unicast switch scheduling in a Banyan network in [6], and in the context of a multicast switch in [7]. This paper extends this idea to the case when fanout splitting

[1]In a hypergraph, a *stable set* is a set of vertices such that no hyperedge is fully covered.

and network coding are allowed at the inputs of a crossbar multicast switch. Through an example we demonstrate that network coding strictly improves the rate region of a multicast crossbar switch. We use the conflict graph idea to derive an outer bound on the rate region of a general multicast crossbar with network coding.

The rest of the paper is organized as follows. Section II gives the formal description of the conflict hypergraph representation of a network coding problem along with an example. Section III contains a discussion on the pros and cons of the new formulation. In Section IV we present examples that show that network coding strictly improves the rate region of a multicast switch. We use the conflict graph approach to give an outer bound on the rate region. Finally, Section V gives the conclusions.

II. The Conflict Hypergraph

We use the same network model as in [2]. We are given a directed acyclic graph $G = (V, A)$ representing the network. The conflict hypergraph corresponding to a network coding problem is defined as follows.

Vertices: Define a set of vertices for each link in A - one vertex for each possible "composition of information" on that link. The composition of information on a link is the net transfer function from the source messages to the symbol sent on the link. For example, consider a binary linear code. The composition of information on a link refers to the set of sources which have been XORed to form the symbol on the link.

Edges: In a valid code, more than one vertex cannot be chosen corresponding to each link. Hence we define an edge between every pair of vertices that represent the same link. Moreover, the composition on an outgoing link at any node in V is valid only if it is a function of the incoming compositions. Thus, the set of possible compositions of outgoing links becomes restricted once we specify the composition of the incoming links. This constraint is modeled using a set of hyperedges. If the composition on an outgoing link at a node is incompatible with a set of incoming input compositions, then the corresponding vertices are connected by a hyperedge.

Under such a construction, a selection of vertices, one corresponding to each link, gives a code. For a code to be valid, the set of vertices must form a stable set, *i.e.* no hyperedge may be fully covered by the chosen vertices. Thus, a valid code has a one-to-one correspondence to a stable set in which there is exactly one vertex corresponding to each link in the network. In fact, any point in the convex hull of such stable sets is achievable through time-sharing.

Fig. 1 shows an example network with a traffic demand of 1 unit from s_1 to t_1 and 1 unit from s_2 to t_2. The example is somewhat similar in structure to the example in [5]. All links have unit capacity. It is assumed that nodes with only one incoming edge simply forward whatever they receive on all outgoing links. Fig. 2 shows the corresponding conflict hypergraph for a scalar linear code over $GF(2)$. The two numbers in parentheses are indicators for which of the two

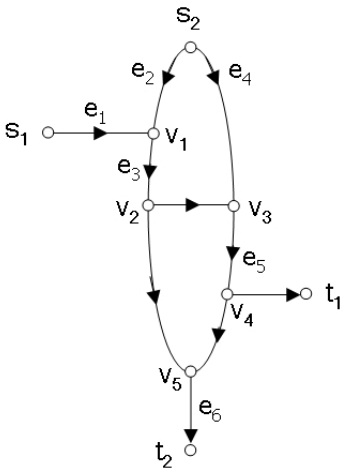

Fig. 1. Example network

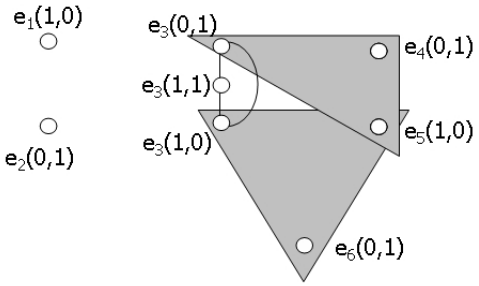

Fig. 2. Conflict graph corresponding to the example

sources are present in the composition of information on that link. For instance, $e_3(1,1)$ means link e_3 carries the XOR of both sources. Note that the only stable set that has a vertex corresponding to every link in the network is $\{e_1(1,0), e_2(0,1), e_3(1,1), e_4(0,1), e_5(1,0), e_6(0,1)\}$. This gives a valid code with nodes v_1, v_3 and v_5 XORing their inputs while the other nodes simply forward the incoming information. The given network is thus solvable over $GF(2)$.

III. Pros and Cons of the Conflict Graph Representation

The conflict graph approach gives a systematic method to find a valid code. Deciding whether a network coding problem has a solution over a given alphabet and class of codes can be reduced to finding whether the conflict hypergraph has a stable set that has exactly one representative vertex for every link in the network. Since the overall problem is represented in terms of local constraints (hyperedges involve only vertices corresponding to links at the same node), the representation scales with the size of the network. The other main advantage of this approach is the flexibility it offers to impose extra constraints on the problem. For instance, if we want to find a solution with network coding only at certain nodes in the network, then we can modify the conflict graph accordingly by

restricting the output composition vertices at the other nodes, and choose a stable set in the resulting hypergraph.

A potential problem with this approach is that the size of the conflict graph may not scale well with the field size used even for linear codes, since the number of possible compositions on a link grows exponentially with the field size. Also, the general problem of finding a stable set with the required properties may be a hard one in some cases. Structural properties of the conflict graph will have to be utilized to simplify the task of finding such a stable set. This is part of future work.

IV. Conflict Graphs and the Multicast Switch

This section aims to demonstrate the application of the conflict graph idea in the context of a crossbar multicast switch. First we summarize the result given in [7]. Then we describe examples to show the benefit of network coding in the switch, in terms of the rate region. Finally, we use the conflict graph idea to derive an outer bound on the rate region of a general multicast crossbar switch with intra-session network coding.

The result in [7] gives a graph theoretic characterization of the rate region of a multicast crossbar switch, under the assumption that fanout splitting[2] is not allowed. The basic idea is to represent the multicast traffic pattern in the switch using a conflict graph. A traffic pattern consists of several multicast flows. (The term "flow" will be used to denote any set of packets with the same input and same set of destinations in the switch) Each flow in the switch is represented by a vertex in the conflict graph, and two vertices are connected by an edge, if the corresponding flows cannot be scheduled simultaneously in the switch. This could be due to a conflict on the input or the output side. Under this model, a valid switch configuration corresponds to a stable set (a set of vertices, no two of which are connected to each other) in the conflict graph. Any switch schedule can be viewed as a time sharing between valid switch configurations. In the conflict graph picture, this corresponds to a convex combination of stable sets. As a result, the achievable rate region with no fanout splitting is the stable set polytope of the conflict graph.

In this section, we wish to understand whether network coding improves the achievable rate region. By network coding, we mean that the inputs of the switch are allowed to code over the packets that are waiting in the queue. We present an example to show that network coding, even when restricted to packets of the same flow, can improve the rate region.

Consider the traffic pattern T shown in Fig. 3. This is a 3×3 switch, with 4 flows – one multicast flow from input 1 to all 3 outputs, and 3 unicast flows from input 2 to outputs 1, 2 and 3 respectively. The rates of the 4 flows are set at $\frac{2}{3}$, $\frac{1}{3}$, $\frac{1}{3}$ and $\frac{1}{3}$ respectively[3].

This traffic pattern cannot be served without fanout splitting. The reason is as follows. The unicasts cannot be served while the multicast is being served. If the multicast flow occupies two-thirds of the schedule, that leaves only one third of the time for the three unicasts. But each unicast needs a rate of $\frac{1}{3}$. This cannot be achieved since they cannot be served simultaneously.

[2]Fanout splitting means serving a multicast packet over several time slots, a few destinations at a time, as opposed to transfering them all at once.

[3]These rates are normalized with respect to the arrival rates

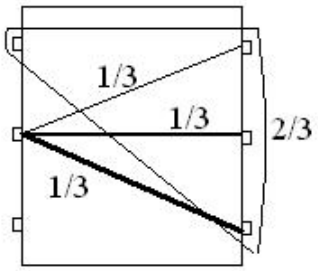

Fig. 3. The example traffic pattern showing the benefit of network coding

Time Slot	Code	Outputs
1	P_1	1,2
2	P_2	2,3
3	$P_1 \oplus P_2$	3,1

TABLE I

The network code used by input 1 for the multicast flow

This traffic pattern cannot be achieved even when fanout splitting is allowed. To show this, we note that at all times in the schedule, one of the unicasts from input 2 has to be served, since it is a saturated input ($i.e.$ the total inflow is 1). Hence, in any time slot, the multicast packet can be sent to at most 2 outputs. Thus, for every packet, there has to be at least one more time slot where it is sent to the output(s) that have not received it. In other words, every packet of the multicast flow will use up at least 2 time slots, implying that a rate of more than $\frac{1}{2}$ is not achievable. Since the required rate is $\frac{2}{3}$, fanout splitting cannot achieve the given traffic pattern.

However, the traffic pattern in Fig. 3 can be achieved if intra-flow network coding and fanout splitting are allowed. In fact, a code over the binary field is sufficient. This is explained next.

A. The Network Code that Solves the Example

The network code involves coding at input 1, over packets only from the multicast flow. Input 1 codes over blocks of 2 packets, and sends them over 3 time slots. For instance, consider a block of packets $\{P_1, P_2\}$ from the multicast flow. The packets in a block are coded in various ways, and sent to different combinations of outputs, in each time slot. The code is described in Table I. All operations are over $GF(2)$. The $\oplus$ sign indicates that the packets are XORed bitwise and sent.

It can be verified that this code enables each of the three destinations to decode both packets in the block, at the end of 3 time slots. For instance, output 1 receives P_1 and $P_1 \oplus P_2$. From these, P_1 and P_2 can be recovered. Thus, each output receives 2 packets of the multicast flow, every 3 slots. This

Flow id	Input	Fanout Set
1	1	1,2,3
2	1	2,3,4
3	2	1,3
4	2	2,4
5	2	1,4

TABLE II

AN EXAMPLE THAT REQUIRES CODING ACROSS FLOWS

	Input 1	Input 2
Time Slot 1	$P_1 \rightarrow \{1,3\}$	$P_4 \rightarrow \{2,4\}$
Time Slot 2	$P_2 \rightarrow \{2,4\}$	$P_3 \rightarrow \{1,3\}$
Time Slot 3	$P_1 \oplus P_2 \rightarrow \{2,3\}$	$P_5 \rightarrow \{1,4\}$

TABLE III

THE NETWORK CODE USED TO SATISFY THE PATTERN IN TABLE II

means, an average rate of $\frac{2}{3}$ has been achieved.

As for the unicast flows, they are also served in these time slots, in parallel. Note that, input 1 talks to only 2 outputs at any given time (column 3 of the table). Input 2 uses this fact to send a unicast packet to the third unoccupied output. For instance, input 2 can talk to output 3 in time slot 1. In this manner, input 2 conveys one unicast packet to each output in every 3 slots. Thus, a rate of $\frac{1}{3}$ is achieved for each of the unicasts. In other words, the given code satisfies all the rate requirements of the example.

B. An Example that Requires Inter-Flow Coding

Whereas the above example shows that intra-session coding improves the rate region, we now present another example traffic pattern that requires inter-flow coding to be satisfied. This example is in a 2×4 switch, with 5 multicast flows. The flows are described in Table II. The required rate for each flow is $\frac{1}{3}$.

In any schedule that achieves a rate of $\frac{1}{3}$ for each of these flows, it is easily seen that the last 3 flows have to be served at different times (since they are from the same input), and their fanout cannot be split. If inter-flow coding is not allowed, then it is not possible to accommodate the first two flows into this schedule since that would require input 1 to send out information from two different flows at once. However, if inter-flow coding is allowed, then the required rate can be achieved. The schedule that achieves the required rate is shown in Table III. In this table, P_i denotes the packet of the i^{th} flow in Table II.

C. An Outer Bound on the Rate Region of a General Multicast Switch

To derive an outer bound on the rate region of a general multicast switch with fanout splitting and intra-flow network coding, we define the *enhanced conflict graph* as follows. If fanout splitting is allowed, then, one possible approach is to think of a multicast flow as being made up of several sub-flows. Hence, for a multicast flow with a fanout of size f, define f vertices, one for each sub-flow. Conflict edges are defined between every pair of sub-flows originating at the same input or terminating at the same output, except that sub-flows belonging to the same flow are not connected to each other.

Theorem 1: The stable set polytope of the enhanced conflict graph (suitably projected from sub-flows to flows)[4] is an outer bound on the rate region with fanout splitting and intra-flow network coding.

Proof: Let **r** be any achievable rate vector with one entry for each flow. Let **r'** be the corresponding rate vector with one entry for each sub-flow – where all entries for sub-flows of the same flow are equal to the rate of that flow in **r**. Since **r** is within the rate region, there exists a schedule that achieves it. In other words, there is a sequence of switch configurations and associated codes for each time slot in the schedule such that the average over time of the rate served for each sub-flow equals the required rate for the corresponding flow. More formally, given the achieving schedule, define an indicator vector in each time slot, with one entry for each sub-flow. This entry has a 1 for those sub-flows in which a new degree of freedom is conveyed by the code in that time slot. Then **r'** is the time average of such indicator vectors over all the time slots. But then, each indicator vector is the incidence vector of some stable set of the enhanced conflict graph. Thus, any achievable rate vector can be expressed as a convex combination of stable sets of the enhanced conflict graph, and this proves the theorem. ■

D. Network Coding Region is Within the Admissible Region

For any multicast traffic pattern, a trivial outer bound on the rate region is that the total rate of all flows destined for any output in the switch must be at most 1. This condition is called the *admissibility* condition. It is interesting to note that there are traffic patterns that are admissible, but cannot be achieved even with inter-flow network coding in a multicast switch. One such example is given here. Consider the traffic pattern shown in Fig. 4. Clearly, no output is oversubscribed and therefore, the pattern is within the admissible region. However, this traffic pattern cannot be supported even if inter-flow network coding is allowed. To show this, we first note that if fanout splitting is not allowed, then after serving the multicast flow, we have only $\frac{1}{3}$ time for the two unicast flows from input 2. But these flows require a rate of $\frac{1}{3}$ each, which cannot be supported since they themselves are in conflict. Now, suppose fanout splitting is allowed. Even then, the multicast flow can never be split, since input 1 is saturated and splitting the multicast flow will not allow enough time to serve all the incoming flow at input 1. This problem remains even if intra-flow network coding is allowed. Note that, no input has 2 flows going to the same output. Hence, coding across any two different flows at an input will "poison" both flows, with some unnecessary flow's packet. To undo the poisoning, extra time has to be spent to send a "remedy". But, since the outputs are already saturated, there is no time for a remedy flow to be sent. Therefore, inter-flow network coding will also

[4]Note that the enhanced conflict graph has one vertex for each sub-flow. To project the stable set polytope onto flows, we need to impose the condition that the rates of all sub-flows of the same flow are equal.

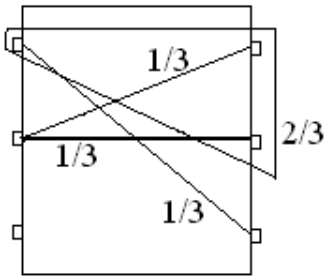

Fig. 4. An admissible traffic pattern that cannot be achieved even with inter-flow network coding

not help. The problem here is that, even if we allow coding across flows, we are still restricted to code only over flows at the same input. This example implies that although network coding does improve the rate region, it cannot achieve the entire admissible region. Reference [11] gives an example of an admissible traffic pattern that cannot be achieved even with fanout splitting. In the same example, even if network coding is allowed in addition to fanout splitting, the traffic cannot be served. Hence, this is yet another example for the fact there are admissible traffic patterns that cannot be sustained even with network coding.

V. Conclusion

In this paper we have introduced a new graph-theoretic formulation of the non-multicast network coding problem using the concept of conflict graphs. This formulation could provide a systematic approach to the problem of deciding solvability of a given network, over a given field and a given class of codes. The approach has its own pros and cons. In particular, it can easily incorporate extra constraints on the problem by suitably modifying the conflict graph.

We have also demonstrated the use of the conflict graph idea in the context of a multicast crossbar switch. We have shown that allowing network coding at the inputs of a crossbar switch can provide benefits in the rate region and have derived an outer bound on the general rate region when fanout splitting and network coding are allowed using the conflict graph.

Acknowledgment

This work is supported by the following grants: NSF Career grant CCR-0093349, NSF ITR grant CCR-032595-1, ONR grants N00014-05-1 and N00014-05-0197, and the AFOSR contract titled “Robust Self-Authenticating Network Coding”.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, “Network information flow,” *IEEE Trans. on Information Theory*, vol. 46, pp. 1204-1216, 2000.

[2] R. Koetter, M. Médard, “An Algebraic Approach to Network Coding”, *IEEE Trans. on Networking*, October 2003.

[3] S.-Y. R. Li, R. W. Yeung, N. Cai. “Linear network coding”. *IEEE Trans. on Information Theory*, February 2003.

[4] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, L. Tolhuizen, “Polynomial time algorithms for multicast network code construction”, *IEEE Trans. on Information Theory*, submitted July 2003, accepted for publication.

[5] S. Riis, “Linear versus Non-Linear Boolean functions in Network Flow”, *Proceedings of the 38th Annual Conference on Information Sciences and Systems*, Princeton, NJ, 2004

[6] C. Caramanis, M. Rosenblum, M. X. Goemans, V. Tarokh, “Scheduling Algorithms for Providing Flexible, Rate-Based, Quality of Service Guarantees for Packet-Switching in Banyan Networks” *Proceedings of the 38th Annual Conference on Information Sciences and Systems*, Princeton, NJ, pp. 160-166, 2004

[7] J. Sundararajan, S. Deb, M. Médard, “Extending the Birkhoff-von Neumann Switching Strategy to Multicast Switches”, *Proc. of the IFIP Networking 2005*, May 2-6 2005, Waterloo, Canada.

[8] T. Ho, M. Médard, J. Shi, M. Effros, D. R. Karger, “On Randomized Network Coding”, *41st Annual Allerton Conference on Communication Control and Computing*, Oct. 2003.

[9] C. Fragouli, E. Soljanin, “Decentralized Network Coding”, *IEEE Information Theory Workshop*, San Antonio, Oct 25-29, 2004.

[10] N. Harvey, D. R. Karger, K. Murota. “Deterministic Network Coding by Matrix Completion”, *ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, Canada, Jan. 2005.

[11] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri: “Multicast traffic in input-queued switches: optimal scheduling and maximum throughput.” *IEEE/ACM Trans. on Networking* Vol. 11, No. 3, June 2003, pp. 465-477.