Network Coding for Wireless Networks

Yunnan Wu

Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399.
yunnanwu@microsoft.com

July 2007

Technical Report
MSR-TR-2007-90

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Network Coding for Wireless Networks

Yunnan Wu
Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399 USA
yunnanwu@microsoft.com

Abstract

Network coding refers to a scheme where a node is allowed to generate output data by mixing (i.e., computing certain functions of) its received data. The unique characteristics of wireless medium renders network coding particularly useful. For instance, network coding can be used to achieve the minimum energy-per-bit for multicasting in a wireless ad hoc network. In addition to optimizing energy efficiency, the network coding based scheme has only polynomial time complexity, breaking through the NP-hardness barrier of the conventional routing approach. As another example, recently network coding has been developed into a link layer enhancement scheme. The network coding engine in the link layer can opportunistically mix the outgoing packets to reduce the transmissions in the air. This paper provides an overview of some recent development about using network coding in wireless networks, including (i) network coding for end-to-end multicasting, (ii) network coding in the link layer, and (iii) network coding in the physical layer.

1 Introduction

In today's practical communication networks such as the Internet and wireless networks, information delivery is performed by routing, i.e., having intermediate routers store and forward data. *Network coding* is a recent generalization of routing in which nodes can generate output data by encoding (i.e., computing certain functions of) previously received input data. As illustrated by Figure 1, in network coding, each node in a network can perform some computation, whereas in routing the output messages can only be copies of the received messages. Intuitively, network coding allows information to be "mixed" at a node.

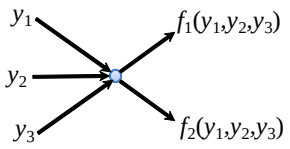


Figure 1: The concept of network coding: Network nodes can compute functions of input messages

The potential advantages of network coding over routing include resource (e.g., bandwidth and power) efficiency, computational efficiency, and robustness to network dynamics. As shown by the pioneering work of Alswede *et al.* [1], network coding can increase

the possible network throughput, and in the multicast case can achieve the maximum data rate theoretically possible. This is illustrated by Figure 2. Each link in the graph can carry one bit per second. Using network coding, we can multicast information from the source node s to the two receivers t_1 and t_2 at rate 2.0 bit/second, which cannot be achieved using traditional routing.

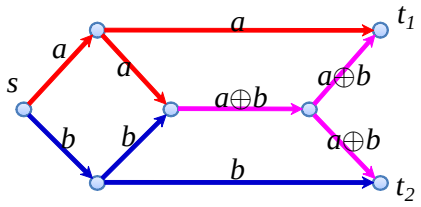


Figure 2: An example showing that network coding can achieve the multicast capacity whereas routing cannot. Here $a \oplus b$ stands for the bitwise XOR of the two bits. This example was introduced by Ahswede *et al.* [1].

In addition to maximizing throughput, network coding can also maximize the energy efficiency. Consider the example wireless network shown in Figure 3. Assume each node is equipped with a transmitter operating at a fixed transmission range, which is just sufficient to reach its lateral neighbors, but not the diagonal ones. Under this setting, each physical-layer transmission consumes a unit amount of energy. Using routing, the minimum number of transmissions required to deliver 1 message from s to $\{t_1, t_2\}$ is 5; a solution is shown in Figure 3(a), where the first transmission is a broadcast transmission. As shown in Figure 3(b), using network coding, we can multicast 2 messages using 9 transmissions, resulting in a better energy efficiency. More generally, for multicasting in a multi-hop wireless network, it has been shown that under a layered model of wireless networks, the minimum energy-per-bit can be found by a linear program; the minimum energy-per-bit can be attained by performing linear network coding [2]. In contrast, minimum energy multicast routing is NP-hard to compute and may not achieve the minimum possible energy-per-bit.

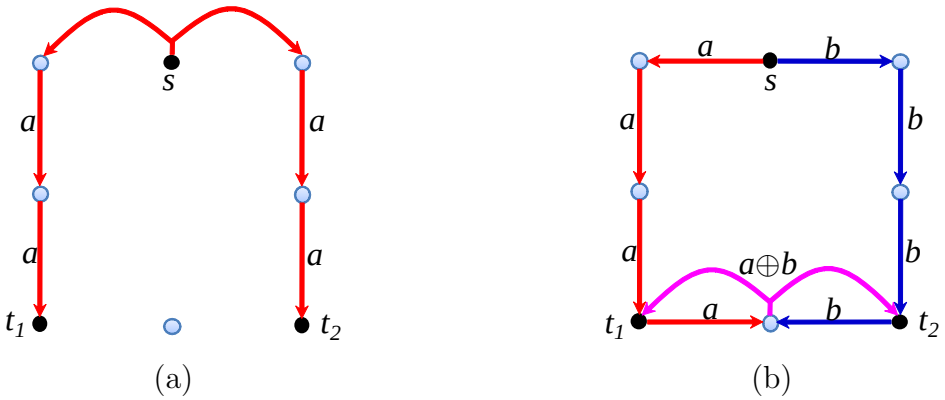


Figure 3: (a): Minimum energy routing solution uses 5 transmissions to multicast 1 message. (b): Minimum energy network coding solution uses 9 transmissions to multicast 2 messages. (Adapted from [3].)

The advantages of network coding go beyond multicasting. Consider the example shown in Figure 4, where the left node wants to send packet a to the right node and

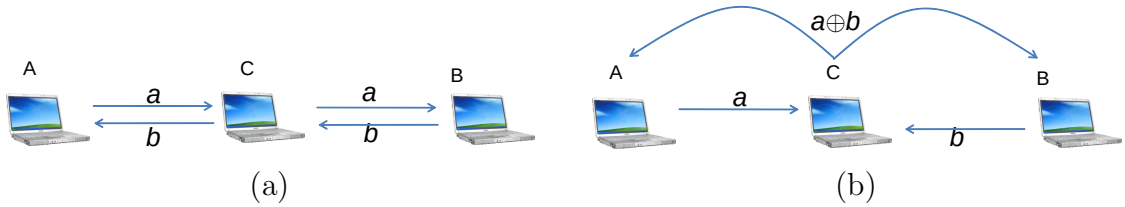


Figure 4: (a) The conventional solution requires 4 transmissions to exchange two packets between A and B via a relay node C. (b) Using network coding, two packets can be exchanged in 3 transmissions.

the right node wants to send packet b to the left node. Using conventional routing, this requires 4 transmissions, as shown in Figure 4(a); using network coding, two packets can be exchanged using 3 transmissions [3]. It looks as if the two packets a and b are sharing a ride in the air. Consider the extension of this scenario to a chain of nodes, where the nodes at the two endpoints want to exchange packets via the intermediate nodes. As the number of nodes approaches infinity, the gain over routing in terms of resource consumption approaches 2:1 [3]. The technique has been further generalized and developed into a generic technique in the link layer of the protocol stack, which mixes packets belonging to different communication sessions.

In the above we have seen some examples that demonstrate the usefulness of network coding in wireless networks. Indeed, in recent years, significant progress has been made regarding the theory and practice of network coding in wireless networks. This paper provides an overview of some recent development. We start by discussing the use of network coding for end-to-end multicasting in Section 2. Then we explain how network coding can be applied as a link layer technique in Section 3. Section 4 reviews some latest results about applying network coding in the physical layer.

2 Network Coding for End-to-End Multicasting

2.1 Theory

In this section we present a theoretical model for analyzing the performance of multicasting information in wireless networks using network coding. We cast this problem as a mathematical optimization, where the objective function is some function of the end-to-end multicast throughput and the overall resource (e.g., energy) consumed in providing such throughput. For simplicity, we assume there is only a single multicast session in the network, from a source node s to a set of destination nodes T.

To establish such an optimization, we use a popular layered model of wireless networks (see, e.g., [4]), which is a mathematical abstraction of the well-known layered network architecture. The basic assumptions of the layered model can be explained as follows. The lower and upper layers are respectively abstracted as supply and demand of communication resource. The interface between the supply and demand is a network of lossless channels with rate limits, which can be described as $G = (V, E, \boldsymbol{c})$ where V and E are sets of vertices and edges respectively and $\boldsymbol{c}$ is a length-$|E|$ vector assigning to each edge $e \in E$ a bit-rate limit $c(e)$.

The physical and link layers can supply many possible graphs $G = (V, E, \boldsymbol{c})$, by applying communication mechanisms such as scheduling, modulation, and channel coding

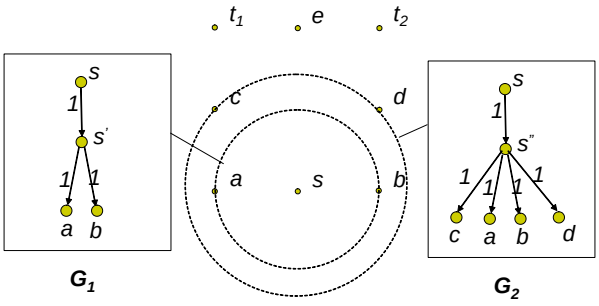


Figure 5: Two example elementary graphs G_1 and G_2. G_1 corresponds to the physical state where s is transmitting at a power just enough to reach a and b. G_2 corresponds to the physical state where s is transmitting at a power just enough to a, b, c, d. (From [3].)

over the underlying noisy and interfering channels. We can these graphs *realizable graphs* because they correspond to feasible ways of operating the wireless network, e.g., arranging different subsets of nodes to communicate for different time fractions. In subsection 2.2, we discuss how to obtain the realizable graphs.

Using a realizable graph G as the available communication resource, the network layer coordinates the information flow from the source to the destinations such that certain end-to-end throughput is achieved. In subsection 2.3, we discuss how to characterize the achievable end-to-end throughput via network coding.

Each realizable graph $G = (V, E, \boldsymbol{c})$ provides certain end-to-end throughput and has an associated resource (e.g., energy) consumption that represents the cost in supplying the bit-rate resources. Putting them together, we obtain an optimization that jointly optimizes the supply side and the demand side. Abstractly, the optimization looks like the following:

$$\begin{aligned} \text{maximize} \quad & U(r) + \lambda p, && (1)\\ \text{subject to:} \quad & G \text{ can provide multicast rate } r; && (2)\\ & G \text{ is a realizable graph with power consumption } p. && (3) \end{aligned}$$

In subsection 2.3.3 we shall concretely examine some optimization formulations for wireless networks.

2.2 Realizable Graphs for Wireless Networks

In this section, we discuss the structure of realizable graphs for wireless ad hoc networks. The set of realizable graphs represents all possible supplies of bit-rate resources arising from power control and scheduling in the physical and medium access layers (under certain simplifying assumptions about the physical layer). For simplicity, we assume the nodes are static and the link conditions do not change over time.

A wireless ad hoc network can operate in many different *physical states*, where each physical state represents a "snapshot" of all nodes in the physical layer, such as which nodes are transmitting, what transmitting powers are being used, and what the channel conditions are. A physical state may support a collection of concurrent *links*, which are assumed to be point-to-multipoint in general. Let V_0 denote the set of nodes in the network. A *link* can be described as $u \xrightarrow{c} Y_u$ where $u \in V_0$ is the transmitter, $Y_u \subseteq V_0$ is its receiver set, and c is the associated bit-rate in a reliable communication.

Each collection of links supported by a certain physical state corresponds to an *elementary (realizable) graph*. Loosely speaking, an elementary graph refers to a graph that can be directly realized in the physical layer. For example, Figure 5 shows two physical states and the corresponding elementary graphs. Figure 5 corresponds to the special case of omnidirectional transmissions. In a first state shown in Figure 5, only node s is transmitting, and the transmission power is just enough to reach a and b. This supports the link $s \stackrel{1.0}{\rightarrow} \{a, b\}$. The corresponding elementary graph is shown as G_1. In a second state, only node s is transmitting, and the transmission power is just enough to reach a, b, c, and d. This supports the link $s \stackrel{1.0}{\rightarrow} \{a, b, c, d\}$. The corresponding elementary graph is shown as G_2.

Generally, an elementary graph consists of a number of concurrent broadcast links. To reflect the broadcasting of common information, we use the following graph model for broadcast links. For each link $u \rightarrow Y_u$, we add to the associated elementary graph a *distinct*[1] virtual vertex (e.g., u'), and unit capacity edges $uu', u'v, v \in Y_u$. Two examples are given in Figure 5 as G_1 and G_2, in which s' and s'' are the virtual vertices introduced. The virtual vertex plays the role of an artificial bottleneck that constrains the rate of new information going out of the transmitter.

By timesharing among different physical states, it is possible to achieve any convex combination of the elementary graphs. That is, if λ_k is the relative share of time for the elementary graph $G_k = (V_k, E_k, \boldsymbol{c}_k)$, then it is possible to achieve on average the graph $G = (\cup_k V_k, \cup_k E_k, \sum_k \lambda_k \boldsymbol{c}'_k)$. Here the edge capacities $\boldsymbol{c}_k$ are each extended to a length-$|\cup_k E_k|$ vector $\boldsymbol{c}'_k$ in the obvious way. Denote such combinations by $G = \sum_k \lambda_k G_k$.

Let the set of all elementary graphs be $\mathcal{B}_0$. The number of elementary graphs, $|\mathcal{B}_0|$, generally grows exponentially with the number of network nodes. For analytic tractability, we shall identify a limited number of promising elementary graphs that provide a reasonably good span for a specific application. For details on how to do so in a general setting, see, for example, [4]. For the minimum-energy multicast problem, however, we only need to examine a polynomial number (in the number of nodes) of elementary graphs, or corresponding physical states. This is because separating interfering transmissions into different time slots improves the energy efficiency. Since we are focusing on the minimum energy-per-bit, we can restrict our attention to those physical states involving only a single transmitter. It is this fact that results in the polynomial solvability of the minimum-energy multicast problem for a wireless ad hoc network.

With a finite set of elementary graphs $\mathcal{B} \subseteq \mathcal{B}_0$, the set of realizable graphs is

$$\mathcal{G}(\mathcal{B}) = \left\{ G \,\middle|\, G = \sum_k \lambda_k G_k,\ \sum_k \lambda_k \leq 1,\ \lambda_k \geq 0\ \forall k, G_k \in \mathcal{B} \right\},$$

where the dependence on $\mathcal{B}$ is explicitly shown. The power consumption of a composite graph $G = \sum_k \lambda_k G_k$ is $\sum_k \lambda_k p(G_k)$, where $p(G_k)$ is the power consumption of elementary graph G_k. The power consumption reflects one possible metric that measures the cost of providing the bit-rate resources at the physical and medium access layers.

[1]To see why it can be problematic to treat the virtual vertices associated with a same transmitter as the same in different elementary graphs, please refer to [5].

2.3 Characterizing End-to-End Throughput of Network Coding in a Given Graph

In this subsection, we assume we are given a network of lossless links represented by $G = (V, E, \boldsymbol{c})$; the graph can be a particular realizable graph supported by the lower layers. We characterize the achievable end-to-end multicast throughput using network coding in G via a set of linear inequalities.

2.3.1 Unicasting and Max-Flow-Min-Cut

Before talking about multicasting, let us begin by reviewing the results for unicasting from a source node s to a destination node t. We are interested in the *unicast capacity*, which refers to the maximum rate at which s can communicate information to t.

An upper bound of the unicast capacity can be obtained by examining the *s–t cuts*. Given two nodes $s, t \in V$, an s–t cut $(U, \overline{U})$ refers to a partition of the nodes $V = U + \overline{U}$ with $s \in U$, $t \in \overline{U}$. The *capacity* of the cut refers to the sum of the edge capacities for edges going from U to $\overline{U}$. An s–t cut with minimum capacity is called a *minimum s–t cut*. Let $\rho_{s,t}(\boldsymbol{c})$ denote the capacity of a minimum s–t cut for graph (V, E) with link capacities $\boldsymbol{c}$. The significance of an s–t cut comes from the fact that it exhibits a bottleneck for communication from s to t. It is intuitively clear that all information t can get from s must be derived from the information flowing across the cut. Consequently, the maximum rate at which s can transfer information to t cannot exceed the minimum s–t cut capacity $\rho_{s,t}(\boldsymbol{c})$.

A fundamental theorem in graph theory, the Max-Flow-Min-Cut Theorem, shows that the cut bound $\rho_{s,t}(\boldsymbol{c})$ is achievable by routing along parallel paths. To explain the Max-Flow-Min-Cut Theorem, we now review the notion of *flow*.

An *s–t flow* is a nonnegative vector $\boldsymbol{f}$ of length $|E|$ satisfying the *flow conservation constraint*:

$$\text{excess}_v(\boldsymbol{f}) = 0, \quad \forall v \in V - \{s, t\}. \tag{4}$$

where

$$\text{excess}_v(\boldsymbol{f}) \equiv \sum_{e \in Out(v)} f_e - \sum_{e \in In(v)} f_e, \tag{5}$$

is the *flow excess of v*, viz., the amount of incoming traffic less the amount of outgoing traffic for node v. The flow excess is not required to be zero at s and t. The flow excess $\text{excess}_t(\boldsymbol{f})$ at the destination node t is called the *value* of the flow.

A flow from a source to a destination can be decomposed into a sum of several *path-flows* and *cycle-flows*. It turns out that the cycle-flows can be eliminated without affecting the value of the flow. Each path-flow corresponds to a path from the source with the destination with an associated rate. The value of the flow is the sum of the rates carried by the individual path-flows. Thus, a flow prescribes a way for information to be routed from the source to the destination along parallel paths; the communication rate achieved by such a routing scheme is the value of the flow. For an illustration of a flow, please see Figure 6(a).

Let $\mathcal{F}_{s,t}(r)$ denote the set of s–t flows, each with its flow value equal to r. Then

$\boldsymbol{f} \in \mathcal{F}_{s,t}(r)$ if and only if

$$\begin{aligned}
&\boldsymbol{f} \geq \boldsymbol{0}, \\
&\text{excess}_s(\boldsymbol{f}) = -r, \\
&\text{excess}_t(\boldsymbol{f}) = r, \\
&\text{excess}_v(\boldsymbol{f}) = 0, \quad \forall v \in V - \{s, t\}
\end{aligned}$$

Note that the above inequalities are linear in $\boldsymbol{f}$ and r; for this reason, $\mathcal{F}_{s,t}(r)$ is called the *s–t flow polyhedron*. A useful property of $\mathcal{F}_{s,t}(r)$ is its linearity in r, i.e.,

$$\mathcal{F}_{s,t}(r) = r\mathcal{F}_{s,t}(1) \equiv \{r\boldsymbol{f} | \boldsymbol{f} \in \mathcal{F}_{s,t}(1)\}. \tag{6}$$

In order for a flow to correspond to a feasible routing arrangement, we often need to enforce that the assigned flow on an edge fits in the available bit-rate resource, i.e., $\boldsymbol{f} \leq \boldsymbol{c}$ (in the element-wise sense). The Max-Flow-Min-Cut Theorem says that the minimum cut capacity $\rho_{s,t}(\boldsymbol{c})$ is equal to the maximum value of an s–t flow within $G = (V, E, \boldsymbol{c})$. It follows then

$$r \leq \rho_{s,t}(\boldsymbol{c}) \Longleftrightarrow \exists \boldsymbol{f}_t \in \mathcal{F}_{s,t}(r),\ \boldsymbol{f}_t \leq \boldsymbol{c}. \tag{7}$$

2.3.2 Multicasting and Edge-Wise Maximum of Flows

Given $G = (V, E, \boldsymbol{c})$, a source node s, and a set of destination nodes T, the *multicast capacity* refers to the maximum multicast throughput. Since the capacity of any s–t cut is an upper bound on the rate at which information can be transmitted from s to t, $\min_{t \in T} \rho_{s,t}(\boldsymbol{c})$ is an upper bound of the multicast capacity. Ahlswede *et al.* [1] showed that $\min_{t \in T} \rho_{s,t}(\boldsymbol{c})$ can be achieved by performing network coding. Hence it is the multicast capacity. Now we have arrived at a characterization of the achievable multicast throughput in a graph $(V, E, \boldsymbol{c})$: End-to-end throughput r can be achieved if and only if:

$$r \leq \min_{t \in T} \rho_{s,t}(\boldsymbol{c}). \tag{8}$$

From (7), we obtain an equivalent formulation of (8): End-to-end throughput r can be achieved in $(V, E, \boldsymbol{c})$ if and only if:

$$\exists \boldsymbol{f}_t \in \mathcal{F}_{s,t}(r),\ \boldsymbol{c} \geq \max_{t \in T} \boldsymbol{f}_t. \tag{9}$$

We call $\max_{t \in T} \boldsymbol{f}_t$ in (9) an *(edge-wise) max of flows*. Just as a flow is the critical structure for unicasting from a source to a destination, a max of flows plays a fundamental role for multicasting using network coding. Figure 6 illustrates the structure of a max of flows, using the classical example of network coding introduced in [1]. Figure 6(a) shows an s–t_1 flow, which prescribes two parallel paths from s to t_1; similarly, Figure 6(b) shows an s–t_2 flow. Figure 6(c) shows the edge-wise maximum of these two flows, which is sufficient to provide a multicast rate of 2.

The "(edge-wise) max of flows" formulation (9) is especially important because it can be written as a set of linear inequalities, and integrated with other constraints in an optimization formulation. We will see some concrete examples in the next subsection.

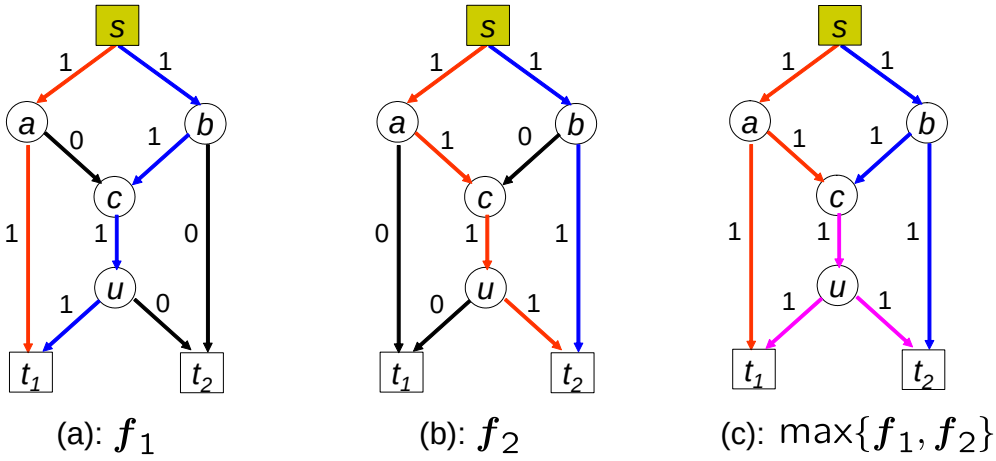


Figure 6: (a) An s–t_1 flow $\boldsymbol{f}_1$ on graph (V, E). (b) An s–t_2 flow $\boldsymbol{f}_2$ on graph (V, E). (c) The edge-wise max of flows $\max\{\boldsymbol{f}_1, \boldsymbol{f}_2\}$, which can provide a multicast rate of 2 from s to $\{t_1, t_2\}$. (From [3].)

2.3.3 Optimization Formulations

Having characterized the supply and the demand sides of wireless networks, we are now ready to formulate various optimizations for network coding–based multicasting in wireless networks. For instance, if we want to find the maximum throughput, we can use the following linear optimization:

$$\begin{aligned}
\text{maximize} \quad & r, && (10)\\
\text{subject to:} \quad & \boldsymbol{c} \geq \boldsymbol{f}_t, \quad \forall t \in T, && (11)\\
& \boldsymbol{f}_t \in \mathcal{F}_{s,t}(r), \forall t \in T, && (12)\\
& \boldsymbol{c} = \sum_k \lambda_k \boldsymbol{c}_k, && (13)\\
& \sum_k \lambda_k \leq 1, && (14)\\
& \lambda_k \geq 0, \quad \forall k. && (15)
\end{aligned}$$

Here we use a finite collection of elementary graphs $\{G_k = (V_k, E_k, \boldsymbol{c}_k)\}$.

As another example, we can minimize the energy-per-bit for multicasting using the following optimization, where r, $\boldsymbol{c}'$, λ'_k, are treated as variables

$$\begin{aligned}
\mathcal{E}^* = \min \quad & \frac{\sum_{k:G_k \in \mathcal{B}} \lambda'_k p(G_k)}{r}\\
\text{subject to:} \quad & \boldsymbol{c}' \geq \boldsymbol{f}_t, \quad \forall t \in T, && (16)\\
& \boldsymbol{f}_t \in \mathcal{F}_{s,t}(r), \forall t \in T, && (17)\\
& \boldsymbol{c}' = \sum_k \lambda'_k \boldsymbol{c}_k, && (18)\\
& \sum_k \lambda'_k \leq 1, \lambda'_k \geq 0, \forall k, && (19)\\
& r > 0. && (20)
\end{aligned}$$

Note that here we only need to include a polynomial number of elementary graphs, each

containing a single transmitter operating at a certain power level (see the discussion in subsection 2.2).

At first glance, the objective function of the above optimization is nonlinear in the variables. However, we can re-normalize the above optimization to arrive at a linear program. Specifically, by a variable change $\lambda_k = \lambda'_k/r$, $\boldsymbol{c} = \boldsymbol{c}'/r$, we have the following linear program:

$$\begin{aligned}
\mathcal{E}^* = \min \quad & \sum_{k:\ G_k \in \mathcal{B}} \lambda_k p(G_k) \\
\text{subject to:} \quad & \boldsymbol{c} \geq \boldsymbol{f}_t, \quad \forall t \in T, && (21) \\
& \boldsymbol{f}_t \in \mathcal{F}_{s,t}(1), \forall t \in T, && (22) \\
& \boldsymbol{c} = \sum_k \lambda_k \boldsymbol{c}_k, && (23) \\
& \lambda_k \geq 0, \ \forall k. && (24)
\end{aligned}$$

2.4 Practice

In this subsection we examine how to practically apply network coding for multicasting in multi-hop wireless networks. Recently, several practical protocols have been proposed [6–9]. There protocols can be understood using a single framework, which contains four pillar ideas: random linear coding, packet tagging, buffering, and output pacing. Random linear coding allows the encoding to proceed in a distributed manner. Tagging each packet with the corresponding coding vector allows the decoding to proceed in a distributed manner. Buffering allows for asynchronous packet arrivals and departures with arbitrarily varying rates, delay, and loss. A proper output pacing mechanism judiciously decides when to generate an output packet, to ensure efficient use of the network resource and coordinate the nodes in moving the information towards the destinations. We now describe them one by one.

Random linear coding: While Ahlswede *et al.*'s work showed that the multicast capacity can be achieved using network coding, their result is mainly an existence proof, established via information theoretical techniques. Hence the first problem is to find a practical way of assigning network codes. The work by Li, Yeung and Cai [10] showed that the maximum multicast capacity can be achieved by using linear encoding functions at each node. With linear encoding, decoding at a receiver amounts to solving linear equations. Linear encoding and decoding represents an important step towards making network coding practical. However, determining the proper coefficients in a distributed manner is another practical challenge. The studies by Ho *et al.* [11] and Sanders *et al.* [12] addressed this challenge, by showing that random linear network coding over a sufficiently large finite field can (asymptotically) achieve the multicast capacity. This leads to a simple and distributed encoding scheme, where each node chooses its own encoding coefficients at random for each output packet, without any coordination with other nodes. Random linear coding is the first key idea for making network coding practical for multicasting.

We now explain the idea of random linear coding in the context of a packet network. For concreteness, suppose $\mathbb{F} = GF(2^8)$ and each packet contains 1000 bytes. Each packet can thus be viewed as a row vector of length 1000, with elements in $GF(2^8)$. Taking Figure 1 as an example, with random linear coding, $f_1(y_1, y_2, y_3) = \alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3$, and $f_2(y_1, y_2, y_3) = \beta_1 y_1 + \beta_2 y_2 + \beta_3 y_3$, where $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3$ are randomly and

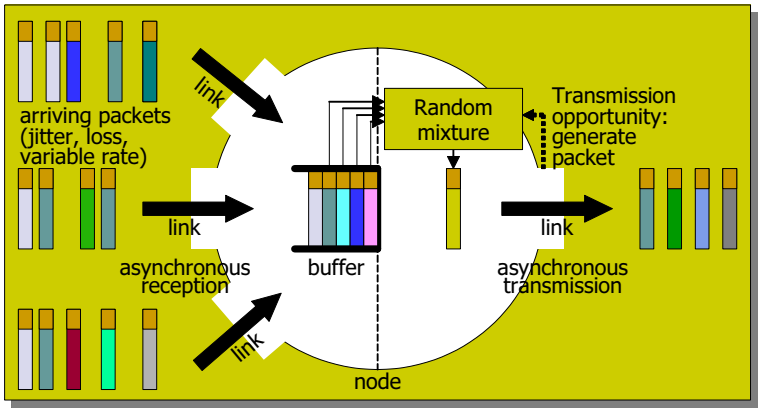


Figure 7: Illustration of the random linear network coding scheme with buffering.

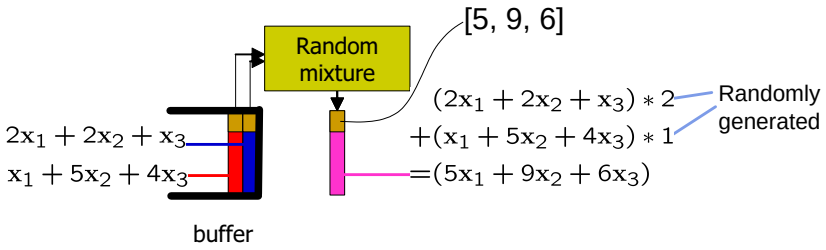


Figure 8: Illustration of the random mixing of packets in the buffer. Suppose $h = 3$ and $\mathbb{F} = GF(11)$. The two packets residing in the current buffer are linear combined with randomly generated coefficients, 2 and 1, respectively. The global coding vector is recorded within each packet to describe how the packet relates to the original packets.

independently drawn from the field $GF(2^8)$. Note that all (1000) symbols within one packet are mixed in the same way.

Packet tagging: Suppose there are h source packets, denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_h$. Linear coding is applied throughout the network so that each packet flowing in the network is a linear combination of the source packets. For example,

$$\mathbf{y} = \sum_{i=1}^{h} q_i \mathbf{x}_i, \tag{25}$$

where $\boldsymbol{q} = [q_1, \ldots, q_h]$ is the *global coding vector* that shows how this packet $\mathbf{y}$ relates to the source packets.

A critical practical issue is to inform the destinations how the coding is done so that they can decode the original packets. To address this issue, Chou *et al.* [13] and Ho *et al.* [11] propose to explicitly record the global coding vector in the packet header. The cost of this scheme is the overhead of transmitting h extra symbols in each packet; note that this is amortized over the number of data symbols in the packet (e.g., 1000). On the other hand, the benefits of the scheme are significant. It allows the system to be completely decentralized: Destinations can decode without knowing the network topology or the encoding rules; destinations can decode even if nodes or links are added or removed in an ad hoc fashion.

Buffering and generations: In many practical networks, synchronization among nodes is difficult and expensive, if not impossible. To make network coding practical, we must handle the asynchronous arrivals of packets and other dynamics. Another important

idea is the use of buffering to eliminate the need for distributed synchronization [13]. As illustrated by Figure 7, each node in the system maintains a buffer. Whenever a node receives a packet from one of its incoming links, it stores the packet into its buffer if the packet is "innovative". A mixture packet is said to be innovative if it is not a linear combination of the packets in the buffer. Non-innovative packets do not provide any new information to the node and hence are immediately discarded. To efficiently test whether a packet is innovative, we can maintain the packets in the buffer in the standard row echelon form. Notice that the testing can be done by using only the global coding vectors of the packets in the buffer, not involving the payloads.

Whenever a transmission opportunity arises on one of its outgoing links, a node generates an output packet by linearly mixing the packets in the buffer using random coefficients in $\mathbb{F}$, as illustrated in Figure 8.

Output pacing: With random linear coding, generating an output packet is easy: Simply combine the buffered packets using a set of randomly generated coefficients. But when should we generate an output packet? This is what we call the "output pacing problem." We now review two example approaches to output pacing.

Widmer, Fragouli, and Le Boudec [6] studied broadcasting in a multi-hop wireless network. In their output pacing protocol, each node maintains a send counter, which is initialized to 0. A new source packet is always broadcast once in the original form (i.e., without mixing with other packets). Each time a node receives an innovative packet, the send counter is incremented by a value called the "forwarding factor" d. A node broadcasts a mixture packet when its send counter is positive. Each time it broadcasts a packet, its send counter is decremented by one. A subsequent paper by the authors [7] extends this scheme by allowing the nodes to have different forwarding factors. Let d_v denote the forwarding factor of v. Here the forwarding factor d_v is adjusted heuristically based on the local (2-hop) density information. A specific rule for setting d_v that works well in the simulations is to set the forwarding factor to be inversely proportional to the number of 1-hop neighbors of v's 1-hop neighbors:

$$d_v = \frac{k}{\min_{v' \in N(v)} |N(v')|}, \tag{26}$$

where k is a network-wide constant parameter, and $N(v)$ is the direct neighbors of node v. The intuition is that if v has a neighbor whose only neighbor is v, then v must forward it, regardless of how many neighbors v has.

Chachulski *et al.* [8] proposed a multicasting protocol called MORE. A unique feature of MORE is its use of distance information. Each node is labelled with the distances to the destinations. Only nodes that are closer to at least one destination than the source are involved in the distribution of the session data. Intuitively, this ensures that information is not flowing in the opposite direction.

Similar to the broadcast protocol discussed above, in MORE, each node performs output pacing by making use of a credit counter. When the MAC allows to transmit a packet, the node generates a mixture when the credit counter is positive, broadcast it, and decrement the credit counter by one. However, a different rule is used for increasing the credit when receiving an innovative packet. When receiving an innovative packet from an *upstream* node, a node increases its credit by `TX_credit`. The `TX_credit` values are determined based on the link loss probabilities in the entire network; for the exact expression, please refer to [8].

Applications: Using network coding, the linear mixture packets can automatically weave its way to the destination if there exists one. Hence network coding is as distrib-

uted, robust, and adaptive as flooding. As a result, network coding–based solutions are attractive candidates for multicasting protocols in multi-hop wireless networks.

Furthermore, the robustness offered by network coding makes it particularly attractive for networks with mobility (e.g., vehicular networks), sensor networks where nodes sleep most of the time to conserve energy, and other scenarios where connectivity is sparse. For these related applications of network coding–based multicasting, please refer to e.g., [14–16].

3 Network Coding in the Link Layer

In Figure 4 we have seen how network coding can be used to reduce the number of transmissions for packet exchanges between two nodes via an intermediate node, leveraging the broadcast medium. The gist of the packet exchange example in Figure 4 is as follows: At certain moment, the left node has a; the right node has b; the middle node has a and b. Thus a mixture packet $a \oplus b$ can be decoded into a and b respectively at the right and left nodes.[2] This packet exchange example bears some similarities with the use of network coding for end-to-end multicasting. Indeed, we can interpret this packet exchange scenario as a virtual multicast session where initially a virtual source node sends packet a to the left node and sends packet b to the right node. Furthermore, the network coding–based information exchange constitutes the critical step in the minimum energy multicast solution, shown in Figure 3(b).

Despite the similarities, this example also has a distinctive feature: the mixture packet can be immediately decoded by the neighbors. The fact mixing and demixing are both done locally brings in many advantages. For instance, a node can mix two packets passing through it, which may belong to different communication sessions; in contrast, in the random linear coding used for end-to-end multicasting, only packets belonging to the same end-to-end session can be mixed together. In fact, the mixing can be done in a way transparent to the communication sessions: They need not know that somewhere in the network, their packets are mixed and then quickly demixed. In addition, a node need not buffer any mixture packet, resulting in a cleaner design that fits the existing networking framework better. To highlight this feature and differentiate with the use of random linear coding for end-to-end multicasting, we use the name "local mixing" to refer to the use of network coding exemplified by Figure 4, where original packets are mixed locally at a node and then demixed immediately at the receivers.

Indeed, this technique has been generalized and developed into a link layer enhancement scheme for multi-hop wireless networks by Katti *et al.* [17]. As illustrated in Figure 9(a), the local mixing engine sits above the MAC layer (e.g., 802.11) and below the network layer. The network layer passes to the local mixing engine a list of packets with their respective next-hops determined according to a certain routing scheme. The local mixing engine maintains information about the packets each neighbor has, and identifies opportunities to mix the outgoing packets to reduce the transmissions in the air. More specifically, each node snoops on the medium and buffers packets it heard. A node also informs its neighbors which packets it has overheard. This allows nodes to know roughly what packets are available at each neighbor (i.e., "who has what?"). Knowing "who has what" in the neighborhood, a node examines its pending outgoing packets and forms

[2] In the following we use the name *source packet* to refer to a packet such as a which was originally generated by a source node, and the name *mixture packet* to refer to a packet such as $a \oplus b$.

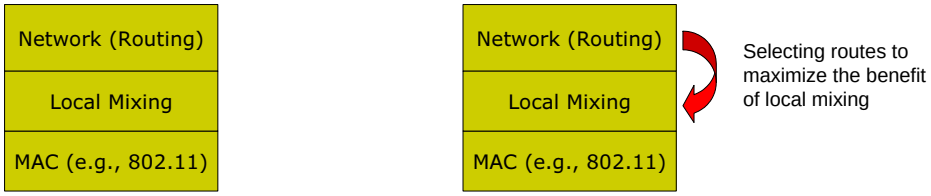


Figure 9: (a) The local mixing engine sits between the network layer and the MAC layer and thus presents an enhanced link layer to the network layer. (b) Local mixing–aware routing schemes can better take advantage of the local mixing engine by generating traffic patterns that have more mixing opportunities.

output mixture packets if possible. In subsection 3.1, we explain the local mixing engine in details.

The local mixing engine, on its own, can improve the link layer efficiency. The gain of this technique, however, critically depends on the traffic pattern in the network. For instance, if we have two flows travelling in opposite directions along a chain, then asymptotically the gain over conventional routing can approach 2:1. In addition, the results of [17] show that the throughput increase can be up to 4x in a multi-hop wireless test-bed for many UDP flows among randomly chosen sources and destinations. The throughput gain is smaller in other traffic patterns (e.g., all traffic are to and from some Internet gateways). This motivates the following question: Can we make intelligent routing decisions that maximize the benefits offered by the local mixing engine? Figure 9(b) illustrates this concept. In subsection 3.2, we review a local mixing–aware routing scheme, proposed in [18].

3.1 Local Mixing

Consider the situation illustrated by Figure 10. A wireless router knows the source packets each neighbor has (i.e., "who has what"). It also knows "who wants what" because these are the packets in its output queue that it is supposed to forward to the neighbors. Then it can decide locally how to optimize the formation of mixture packets. A heuristical approach for generating the mixture packets is used in [17], which takes the packet at the head of the output packet queue, and steps through the packet queue to greedily add packets to the mixture, while ensuring the neighbors can successfully demix. For example, in Figure 10, there are five packets in the output queue, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5$; assume a lower indexed packet is an earlier packet. Then the greedy procedure will use three transmissions: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$, $\boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_5$. However, there is a better solution: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_2 \oplus \boldsymbol{x}_5$. A mathematical abstraction of the optimized formation of the mixture packets – the *local mixing problem* – is studied by Wu et al. [19] from an information theoretic point of view. Under the assumption that each neighbor discards the received packets that are polluted by sources it does not have or want, the optimal mixing is characterized.

Why would a node have packets meant for others? In Figure 4, node v_1 has packet $\boldsymbol{x}_1$ because it is the previous hop of $\boldsymbol{x}_1$. More generally, due to the broadcast nature of the wireless medium, neighboring nodes may overhear packets. For example, in Figure 10, packet $\boldsymbol{x}_1$ may follow a path $\ldots v_4 \rightarrow v_0 \rightarrow v_3 \ldots$; v_1 and v_2 may have overheard $\boldsymbol{x}_1$ when v_4 sent it to v_0.

How does a node get to know "who has what"? First, note that a node can obtain

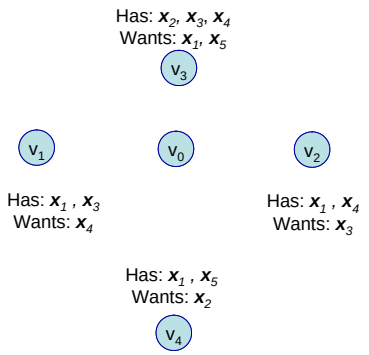


Figure 10: Knowing "who has what" and "who wants what" in a neighborhood, the local mixing engine identifies opportunities to mix the outgoing packets to reduce the resource consumption.

some partial information about its neighbors' data availability in a passive fashion. For example, node v_2 may infer that node v_1 holds packet $\boldsymbol{x}_1$ if v_1 recently received packet $\boldsymbol{x}_1$ or a mixture packet involving $\boldsymbol{x}_1$ from v_1, or if v_2 recently heard v_1 acknowledging the receipt of packet $\boldsymbol{x}_1$. This suffices for packet exchanges such as Figure 4.

Passive inference does not incur any additional overhead. However, using passive inference alone, a node may only obtain a limited view of the neighbors' data availability. Katti et al. [17] extended this by proposing two techniques to obtain more information about local data availability: (i) Let each node explicitly announce the packets it currently has to its neighbors; (ii) let a node guess whether a neighbor has overheard a packet using information about the channel reception. In the former, each node can periodically compose *reception reports* to announce the packets it has overheard. The reception reports may also be piggybacked with ordinary packets. To implement guessing, nodes conduct measurement about the packet success probabilities to its neighbors and exchange the measurement results in the neighborhood. Such measurement and report functionality may already be needed by a routing protocol based on the expected transmission count (ETX) [20]. The guessing technique of [17] can be explained via Figure 10. Suppose v_4 sends a source packet $\boldsymbol{x}_1$ to v_0 without mixing; suppose v_0 knows that v_1 can receive a packet from v_4 with probability 0.8. When v_0 received $\boldsymbol{x}_1$ sent by v_4, v_0 can infer that v_1 has overheard the packet with probability 0.8. Guessing may result in a more up-to-date knowledge about "who has what"; however, if the guess is wrong, the neighbor may fail to demix a packet intended for it. To compensate for the mistakes in guessing "who has what", explicit ACKs can be used. Specifically, nodes can keep track of the packets that were sent but have not yet been acknowledged and retransmit packets after time-out.

3.1.1 Some Implementation Issues

In this subsection, we briefly review the key data structures and operations in a possible implementation.

Each packet has a variable length header that includes: (i) the IDs of the source packets being mixed and their respective receivers, (ii) some piggybacked ACKs, (iii) some piggybacked reception reports. If no data packets were sent after a certain amount of time, then a dedicated control packet containing ACKs and reception reports is broadcast.

Each node maintains three separate buffers, `OverheardBuffer`, `ReceivedBuffer`,

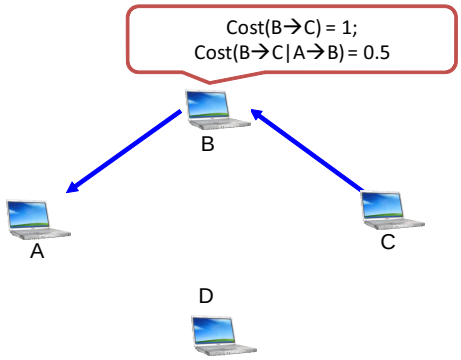


Figure 11: An example mesh networking scenario. Assume currently there is a long-term background flow, $C \to B \to A$. We want to find a good route for a flow from A to C.

`SentBuffer`, holding respectively the source packets that the node overheard, received, or sent. Upon receiving a packet, the packets in these three buffers are used for demixing. Reception reports describe new content in the `OverheardBuffer`. ACKs describe new content in the `ReceivedBuffer`.

Each node maintains a `WhoHasWhatTable` whose entries are of the form "node v_i has source packet $\boldsymbol{x}_j$ with probability p". Upon receiving a packet, the `WhoHasWhatTable` is updated according to the local mixing header. If the received packet is a source packet, guessing is also performed based on the measured channel reception probabilities.

A mixture packet may be intended for more than one receiver. However, the 802.11 protocol has a limited support for MAC-layer broadcast (e.g., broadcast packets are not ACKed). To address this practical issue, Katti *et al.* [17] proposed to use "pseudo broadcast". Specifically, the mixture packet is sent as a unicast packet addressed to one of the receivers. Nodes run in the promiscuous mode to overhear packets; upon receiving a packet, a node inspects the local mixing header to decide whether it is an intended receiver of the packet. A consequence of the pseudo broadcast approach is that the sender cannot be sure whether the other intended receivers received the packet reliably. Such an issue can be addressed by using explicit ACKs, in addition to the ACK in 802.11 MAC [17].

After a packet is sent, the ingredient source packets are moved from the output queue into the `SentBuffer`. In addition, timer events are inserted so that the sent packets will be moved back to the output queue for retransmission if the ACKs does not arrive after a certain time threshold.

3.2 Local Mixing–Aware Routing

Consider the example setting illustrated in Figure 11. There is an existing long-term flow in the network, $C \to B \to A$. We want to find a good route for a flow from A to C. Due to the existence of the local mixing engine, the route $A \to B \to C$ is a good solution because the packets belonging to this new flow can be mixed with the packets belonging to the opposite flow $C \to B \to A$, resulting in improved resource efficiency. But how do we realize that $A \to B \to C$ is a better route than $A \to D \to C$?

Before explaining the local mixing–aware routing solution, we first review how routing is commonly done in wireless mesh networks. Conventionally, routing protocols in wireless mesh networks have been based on finding shortest paths. Here the cost of a path is modelled as the sum of the costs on the constituting links, and the cost of a link

typically reflects the link quality in one way or another. Hence a natural first thought is to modify the link metrics to take into account the effect of the local mixing engine in reducing the transmissions in the air. For instance, can link $B \to C$ announce a lower cost? There are some issues in doing so, because a packet from D that traverses $B \to C$ may not share a ride with a packet from C that traverses $C \to B$, although a packet from A that traverses $B \to C$ can. We see from this example that in the presence of the local mixing engine, assessing the channel resource incurred by a packet transmission requires some context information about where the packet arrives from. For example, we can say that giving the current traffic condition, the cost of sending a packet from B to C that previously arrive from A is smaller, because it can be mixed with the background flow. This observation can be modelled by a *conditional link cost*. Let $\mathsf{cost}(B \to C | A \to B)$ denote the cost of sending a packet from B to C, conditioned on that the packet arrived from A.

Wu *et al.* [18] proposed to use a specific conditional link metric called the ERC (expected resource consumption) to model the resource saving due to local mixing. With the local mixing engine, several packets may share a ride in the air. Naturally, the passengers can share the airfare. In effect, each participating source packet is getting a discount. The ERC metric models the resource consumption while taking such discount into account. Consider a packet sent in the air. If it is a mixture of k source packets, then the ERC metric "charges" each ingredient source packet $1/k$ the resource consumed by the packet transmission. Here the resource consumed by the transmission could be measured in terms of, e.g., air time, or consumed energy. Consider Figure 11. Suppose that the link $B \to C$ has a normal cost of 1. Node B may announce a lower conditional link cost, $\mathsf{cost}(B \to C | A \to B) = 0.5$, to reflect that a new packet from B can enjoy a 50% discount since it can be mixed with the background flow.

Using a set of conditional link costs, the cost of path can be evaluated with a *Markovian metric* [18]. Consider a path $\mathcal{P} = v_0 \to v_1 \to \ldots \to v_k$. A *Markovian metric* models the cost of a path as the sum of the conditional costs on the links:

$$\mathsf{cost}(\mathcal{P}) \stackrel{\Delta}{=} \mathsf{cost}(v_0 \to v_1) + \mathsf{cost}(v_1 \to v_2 | v_0 \to v_1) + \ldots + \mathsf{cost}(v_{k-1} \to v_k | v_{k-2} \to v_{k-1}). \tag{27}$$

The conventional routing metric can be viewed as a special case of the Markovian metric where all the conditional link costs are equal to their unconditional counterparts. The decomposition relation (27) is reminiscent of the decomposition of the joint probability distribution of random variables forming a Markov chain into a product of the conditional probabilities. Thus, a Markovian metric to an unconditional metric is like a Markov chain to a memoryless sequence of random variables. Due to this decomposition structure, the dynamic programming principle still applies and thus finding the shortest path with a Markovian metric can still be done in polynomial time. In a practical network, support for the Markovian metric can be added easily into an existing routing framework that uses a conventional routing metric.

Let us now see some performance results from [18]. The simulation topology is the 9-node grid network scenario shown in Figure 12. Three UDP flows, $v_9 \rightsquigarrow v_1$, $v_1 \rightsquigarrow v_9$, and $v_3 \rightsquigarrow v_1$, are simulated. Each flow begins randomly between 50–60 seconds into the simulation. The results are depicted in Figure 13(a). Three systems, LQSR, LQSR+LM, and MMSR, are compared. LQSR is a link state routing system using a link quality routing metric, which is used as a baseline. LQSR+LM is obtained by adding local mixing support in the link layer. MMSR uses local mixing in the link layer and a local

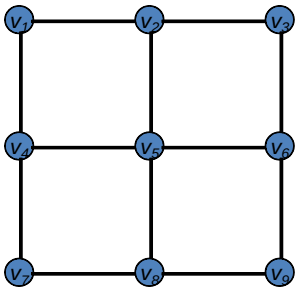
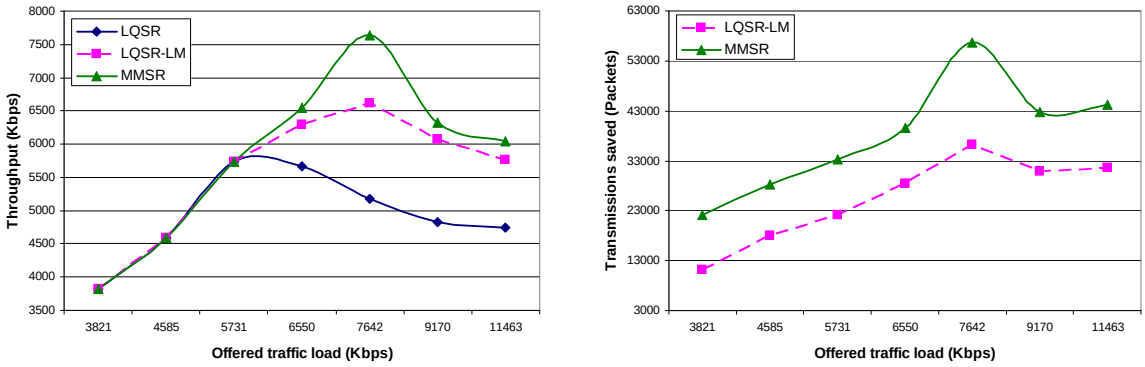


Figure 12: A 9-node grid network.



(a) (b)

Figure 13: (a) Throughput comparison of MMSR, LQSR+LM and LQSR. (b)Transmissions saved through mixing in MMSR and LQSR+LM.

mixing–aware routing system based on Markovian metric. The x-axis stands for the input traffic load. It is observed that LQSR cannot sustain the throughput imposed by the input flows to the network as the load increases. MMSR provides significant throughput gains compared to LQSR (up to 47%) and LQSR+LM (up to 15%). This is because not only does MMSR allow subsequent flows to mix with existing flows, it explicitly tries to maximize mixing. In the example, the flow 1-2-3-6-9 is mixed with 9-6-3-2-1 with MMSR, due to the mutually beneficial discounts enjoyed by both flows. Figure 13(b) gives the amount of *resource* saved by using MMSR; the y-axis is the number of original source packets minus the number of actual transmissions. MMSR consistently provides reduction of packet transmissions of over 10,000 packets across a wide variety of traffic demands.

In summary, the local mixing engine, on its own, can improve the link layer efficiency; it identifies mixing opportunities on the fly and takes advantage of them if they are present. Routing with a Markovian metric makes local mixing more useful as it creates more mixing opprotunities. This can translate to notable resource saving and throughput gain, as confirmed by simulations. For a theoretical (flow-based) analysis of local mixing–aware routing, please see the recent work by Sengupta *et al.* [21].

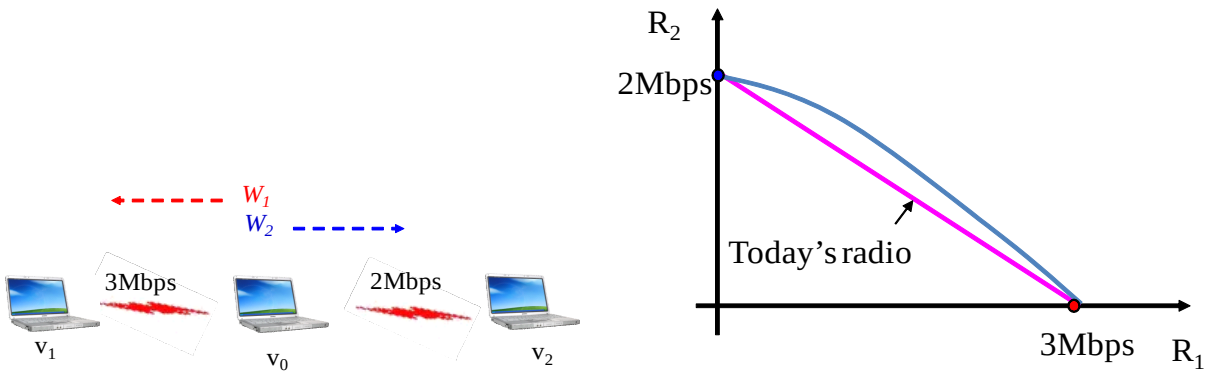


Figure 14: (a) The basic broadcasting scenario, where the node in the middle wants to send a packet a to v_2 and a packet b to v_1. (b) The capacity region of the additive white Gaussian broadcast channel is the upper curve.

4 Network Coding in the Physical Layer

In Sections 2 and 3, the network coding techniques mix the information packets and each packet is either lost or received free of error. In this section we discuss some physical layer mixing techniques that lead to further performance improvement. These techniques share some similarities with the use of mixing at the packet level; these techniques may also be used in conjunction with mixing at the packet level. We loosely classify these mixing techniques as "network coding in the physical layer".

4.1 Mixing at the Modulator/Channel Coder

Consider the basic broadcasting scenario illustrated by Figure 14(a), where node v_0 wants to send a message W_2 to v_2 and a message W_1 to v_1. Suppose v_0 can communicate at a maximum rate of 3Mbps with v_1 and at a maximum rate of 2Mbps. Let R_1 and R_2 denote the long-term rates at which v_0 can send distinct messages to v_1 and v_2, respectively. Hence the pairs $(R_1, R_2) = (3, 0)$ and $(R_1, R_2) = (0, 2)$ (Mbps) are achievable. In today's radio transceivers, node v_0 alternates between transmitting data to v_1 and transmitting data to v_2. Hence, as shown in Figure 14(b), the straight line connecting $(R_1, R_2) = (3, 0)$ and $(R_1, R_2) = (0, 2)$ is achievable by timesharing. Timesharing, however, is not the best possible scheme. For this scenario, it is known in information theory that a larger rate region is achievable. If the v_0 is related to v_1 and v_2 by additive white Gaussian noise (AWGN) channels, then the capacity region, i.e., the set of all achievable rate pairs,is [22–24]:

$$\left\{(R_1, R_2) \mid R_1 \le C\left(\frac{\alpha P}{N_1}\right), \quad R_2 \le C\left(\frac{(1-\alpha)P}{\alpha P + N_2}\right)\right\}, \tag{28}$$

where P is the transmission power at v_0, N_1, N_2 are the noise levels at v_1 and v_2, respectively, $C(\mathsf{SNR}) = \frac{1}{2}\log_2(1 + \mathsf{SNR})$, and α is an arbitrary constant in $[0, 1]$. The capacity region is illustrated by the upper curve in Figure 14(b). To achieve any rate pair in the capacity region, the *super-position coding* technique proposed by Cover [22] can be used. The transmitter generates two codebooks, one with 2^{nR_1} entries and power αP, and the other with 2^{nR_2} entries and power $(1-\alpha)P$. To send messages $W_1 \in \{1, \ldots, 2^{nR_1}\}$ and

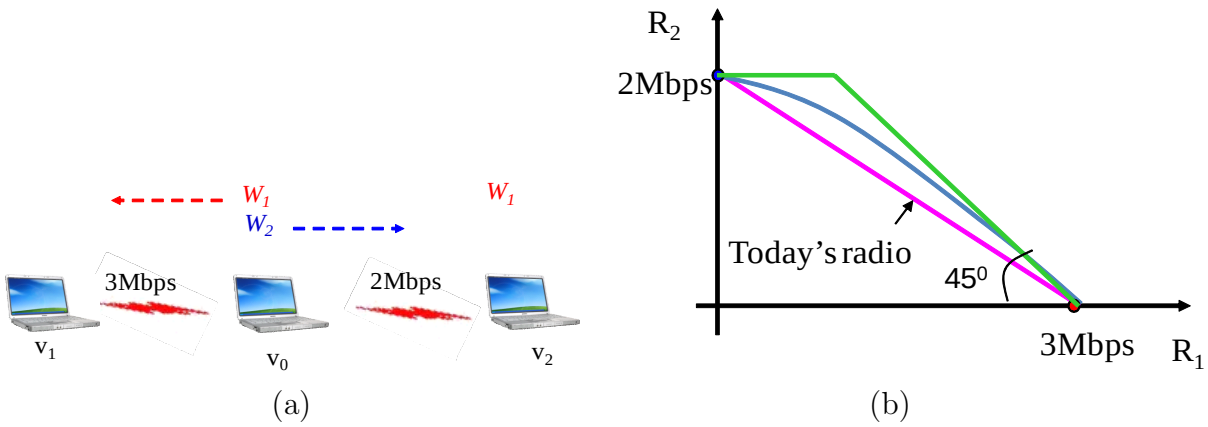


Figure 15: (a) Broadcasting when v_2 knows W_1 a priori (b) The capacity region is enlarged with the side knowledge.

$W_2 \in \{1, \ldots, 2^{nR_2}\}$ to v_1 and v_2, respectively, v_0 sends $\boldsymbol{x}_1(W_1)+\boldsymbol{x}_2(W_2)$, where $\boldsymbol{x}_1(W_1)$ is the codeword indexed by W_1 in the first codebook and $\boldsymbol{x}_2(W_2)$ is the codeword indexed by W_2 in the second codebook. Receiver v_2 decodes W_2 by treating $\boldsymbol{x}_1(W_1)$ as noise. Receiver v_1 decodes W_2 first, then subtracts $\boldsymbol{x}_2(W_2)$ and finally decodes W_1. By varying α, different points in the region can be achieved.

In a practical wireless network, sometimes a receiver may have prior knowledge about some messages destined to other nodes. For example, as illustrated by Figure 15(a), receiver v_2 may know the message W_1 a priori. Such a situation could arise because W_1 originates at v_2 and is then relayed by v_0 to v_1, or v_2 happens to overhear W_1 when some other nodes was sending W_1 to v_0. Interestingly, the presence of the prior knowledge can enlarge the rate region. As illustrated in Figure 15(b), if v_2 knows W_1 a priori, then the capacity region is given by the interior of the uppermost curve, i.e., $\{(R_1, R_2)|R_2 \leq 2, R_1 + R_2 \leq 3\}$ [25–27]. This means when v_0 is sending at the maximum rate (2Mbps) to v_2, it can still pack information to v_1 at 1Mbps for free!

The key constructive technique for achieving the enlarged rate regions in Figure 15 is *nested coding*. Let us first see an informal explanation of this technique. We show how to achieve the point $(R_1, R_2) = (1, 2)$. As illustrated by Figure 16, every microsecond, we encode 3 bits for every symbol via 8-PSK uncoded modulation, where the first two bits are from W_2 and the last bit is from W_1. In this case, receiver v_1 has a good channel with rate 3Mbps; hence it can differentiate the 8 possible constellations and recover all three bits. Receiver v_2 only has a channel with rate 2Mbps, but it knows the last bit of the symbol from its prior knowledge about W_1. Hence to receiver v_2, the transmitted symbol appears to be from a 4-point constellation. Thus it can resolve the ambiguity and receive 2 bits about W_2 every microsecond.

The above is an informal explanation of nested coding. More formally, consider coding over a block of n symbols; in nested coding, the transmitter uses a codebook that can be viewed as 2^{nR_1} sub-codebooks, each having 2^{nR_2} entries. The 2^{nR_1} sub-codebooks are indexed by W_1. To a receiver without prior knowledge, the transmitted codeword appears to be from a codebook of size $2^{n(R_1+R_2)}$. To a receiver that knows W_1 a priori, the transmitted codeword appears tob e from a codebook of size 2^{nR_2}. Such nested coding has been used, for example, by Yang and Høst-Madsen [27], as an efficient scheme for cooperative relaying.

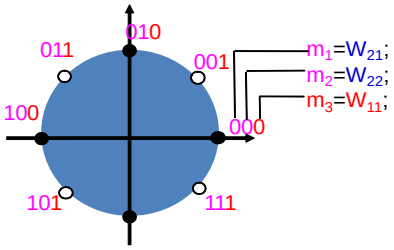


Figure 16: Illustration of the nested coding.

Regarding practical design, Xiao *et al.* [26] recently proposed a practical way of implementing nested coding. The scheme separately encodes multiple information packets via linear channel codes and then computes the XOR of the encoded packets at the physical layer prior to transmission. Specifically, consider mixing several information packets $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_N$ together at the channel encoder. The resulting codeword is:

$$\boldsymbol{c} = \boldsymbol{i}_1\boldsymbol{G}_1 \oplus \boldsymbol{i}_2\boldsymbol{G}_2 \oplus \ldots \oplus \boldsymbol{i}_N\boldsymbol{G}_N, \tag{29}$$

here $\boldsymbol{G}_1, \ldots, \boldsymbol{G}_N$ are different generator matrices. The mixture codeword is then modulated and sent for transmission. When decoding, a receiver that knows some packets can employ the a priori knowledge of some packets to reduce the effective code rate. The contribution of the known packets can be treated as a known scrambling bit pattern; channel decoding can then be modified to account for the scrambling bit pattern. For details about hard- and soft-decision decoding methods for such nested codes, please refer to [26].

Note that the technique does not apply to the dual case where the better receiver v_1 knows W_2 a priori, but v_2 has no side knowledge. Indeed, it has been shown [25] that in such case, the capacity region remains equal to that of the classical broadcast channel.

What if both receiver v_1 knows W_2 a priori and receiver v_2 knows W_1 a priori? Interestingly, in this case, as illustrated by Figure 17, the capacity region is the entire rectangle: $\{(R_1, R_2)|R_2 \leq 2, R_1 \leq 3\}$. So v_0 can simultaneously communicate at the maximum bit-rates to both v_1 and v_2, without raising the power! Let us see how we can do this. As illustrated in Figure 18, we again use 8-PSK to send three bits, m_1, m_2, m_3. But now the first two bits are the XOR of the first two bits of W_1 with the first two bits of W_2. The last bit is the third bit of W_1. Again, node v_1 can recover all three transmitted bits. Then it can use its side knowledge about W_2 to recover W_1. Knowing W_1 a priori, node w_2 can also recover the transmitted bits by resolving the QPSK ambiguity; it can then recover W_2 from the received bits m_1, m_2.

Note that if we apply local mixing at the packet level, we can achieve the point $(R_1, R_2) = (2, 2)$ by XORing the two messages and send them at 2Mbps. Hence by making some changes at the physical layer (e.g., using software-defined radios), the communication efficiency can be improved.

More generally, for two receivers, there could be five types of messages, as illustrated by Figure 19. The 5-dimensional capacity region is characterized in [25]; the capacity region can be achieved by a combination of superposition coding, nested coding, and network coding.

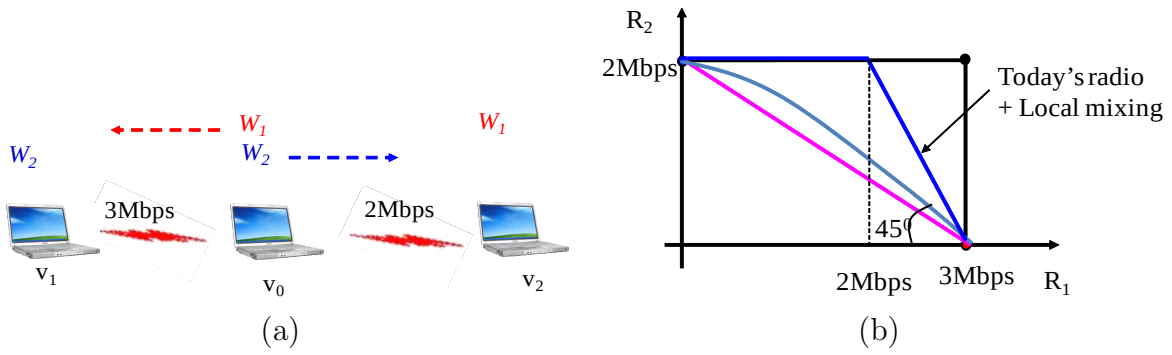


Figure 17: (a) Broadcasting when v_2 knows W_1 a priori and v_1 knows W_2 a priori.(b) The the capacity region is the entire rectangle: $\{(R_1, R_2)|R_2 \leq 2, R_1 \leq 3\}$.

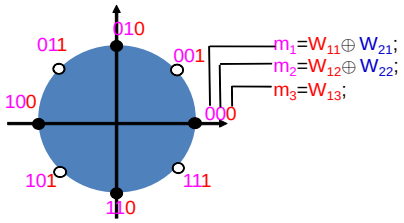


Figure 18: Joint network coding and nested coding.

Receiver \ Message	W_1	W_2	W_3	W_4	W_5
v_1	Wants		Wants	Has	Wants
v_2		Wants	Wants	Wants	Has

Figure 19: For two receivers, there are five types of messages $W_1, \ldots, W_5$.

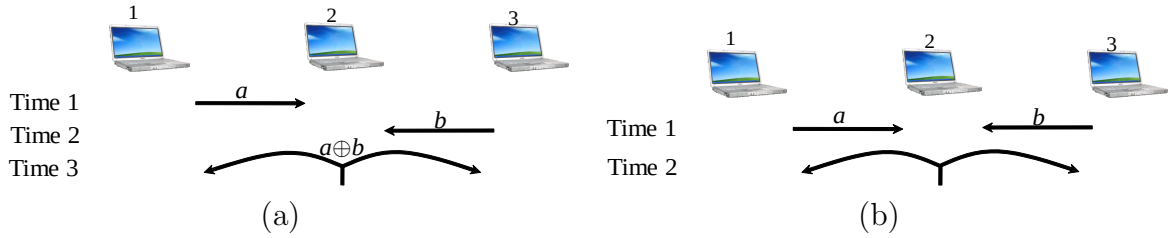


Figure 20: (a) With network coding, two packets can be exchanged in 3 steps. (b) With "mixing in the air", two packets can be exchanged in 2 steps.

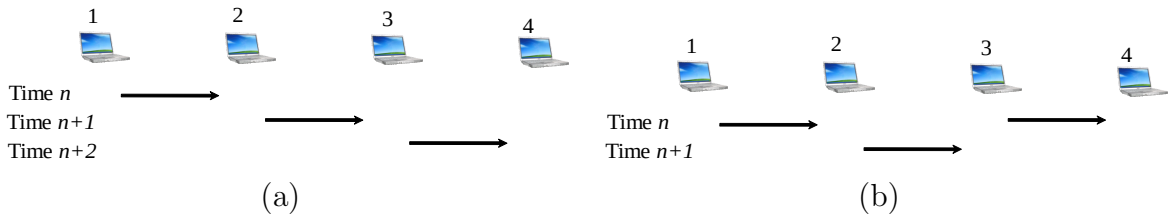


Figure 21: (a) Conventionally, the three links on a path have to take turns. (b) In fact, the first and third link can happen concurrently because node 3's transmission is known interference to node 2.

4.2 Mixing in the Air, Demixing by Cancelling Known Signals

Consider again the example in Figure 4. As shown in Figure 20(a), we can use three transmissions to exchange two packets between node 1 and node 3 via network coding. Recent work [28, 29] noted that it is in fact possible to use two transmissions to exchange two packets. As illustrated in Figure 20(b), in the first time-slot, let nodes 1 and 3 transmit simultaneously. Therefore, node 2 will receive a noisy superposition of the two signals subject to channel distortion, e.g., delay, phase shift. Node 2 can amplify the received noisy superposition and broadcast it. Then the received signal at nodes 1 will be a different noisy superposition of the two signals subject to channel distortion. At a receiver, say node 1, since it knows its original signal a, it should be able to cancel out the contribution due to its own signal and then recover b.

Consider another scenario depicted in Figure 21 [28, 29]. Here we have a 3-hop flow over a chain of 4 nodes. As illustrated in Figure 21(a) Conventionally, the three links have to take turns in transmission. In particular, links $1 \rightarrow 2$ and $3 \rightarrow 4$ cannot be simultaneously active because node 1 and node 3's transmissions collide at node 2. Figure 21(b) shows that it is in fact possible to let node 1 and node 3 transmit concurrently. The critical observation is that node 2 already knows what node 3 is transmitting because it forwarded the packet earlier. Hence, it can cancel node 3's transmission out before decoding node 1's transmission.

The above discussion left out many practical issues, such as compensation of the channel distortion and the time shifts, and necessary changes in the higher layers to work in concert with the advanced physical layer. For more information about how to address them, please refer to [28, 29].

5 Conclusion

An increasingly important application domain of network coding is wireless networks. This paper provides an overview of the theory and practice of network coding in wireless networks.

In Section 2, we started with network coding for multicasting. We showed how to obtain convex optimization formulations for network coding based multicasting in wireless networks. We then discussed the core components of random linear mixing–based multicasting protocols. By having random mixture packets self-orchestrate multiple paths, network coding offers built-in error protection and adaptivity to topology changes due to joins, leaves, node or link failures, congestion, etc; by employing a flooding-type delivery, network coding can be implemented in a distributed fashion easily. These properties render network coding particularly for unicasting and multicasting in mobile ad hoc networks.

In Section 3, we discussed how network coding can be used as a generic technology that improves the link layer efficiency by exploiting the broadcast nature of the wireless medium. The local mixing engine buffers packets and opportunistically mixing packets passing by, to reduce the number of transmissions. The benefit is further increased with the recent development of local mixing–aware routing.

In Section 4, we discussed some recent physical layer techniques that are based on the mixing and demixing of signals. By mixing at the modulator/channel coder and by exploiting the inherent mixing offered by the wireless medium, further performance gains can be achieved.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Communications*, vol. 53, no. 11, pp. 1906–1918, Nov. 2005, also presented at the Information Theory Workshop, San Antonio, TX, Oct., 2004.

[3] ——, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *Proc. 39th Annual Conf. Inform. Sci. and Systems (CISS)*, Baltimore, MD, Mar. 2005, [Online] http://research.microsoft.com/∼yunnanwu. Also available as Microsoft Research Tech. Report, MSR-TR-2004-78, Aug. 2004.

[4] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung, "Network planning in wireless ad hoc networks: a cross-layer approach," *IEEE J. Sel. Areas in Comm.*, vol. 23, no. 1, pp. 136–150, Jan. 2005.

[5] Y. Wu, "Network coding for multicasting," Ph.D. dissertation, Princeton University, Nov. 2005, http://research.microsoft.com/∼yunnanwu/.

[6] J. Widmer, C. Fragouli, and J.-Y. L. Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *Proc. 1st Workshop on Network Coding, Theory, and Applications (NetCod)*, Riva del Garda, Italy, Apr. 2005.

[7] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," in *Proc. INFOCOM*, Barcelona, Spain, Apr. 2006.

[8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2007-014, Feb. 2007.

[9] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard, "Codecast: A network-coding-based ad hoc multicasting protocol," *IEEE Wireless Communications*, pp. 76–81, Oct. 2006.

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

[11] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[12] P. Sander, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Symposium on Parallel Algorithms and Architectures (SPAA)*. San Diego, CA: ACM, June 2003, pp. 286–294.

[13] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Comm., Ctrl. and Comp.*, Monticello, IL, Oct. 2003.

[14] J. Widmer and J.-Y. L. Boudec, "Network coding for efficient communication in extreme networks," in *Proc. ACM SIGCOMM'05 Workshops*, Philadelphia, PA, Aug. 2005.

[15] G. Zhang, J. Neglia, J. Kurose, and D. Towsley, "On the benefits of random linear coding for unicast applications in disruption tolerant networks," in *Proc. 2nd Workshop on Network Coding, Theory, and Applications*, Boston, MA, Apr. 2005.

[16] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla, "Codetorrent: Content distribution using network coding in vanets," in *1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (MobiShare'06)*, Los Angeles, CA, Sept. 2006.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *SIGCOMM*. Pisa, Italy: ACM, Sept. 2006.

[18] Y. Wu, S. M. Das, and R. Chandra, "Routing with a markovian metric to promote local mixing," in *Proc. IEEE INFOCOM 2007 Minisymposium*. Anchorage, Alaska: IEEE, May 2007.

[19] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Proc. Information Theory and Applications Workshop*. San Diego, CA: Univ. of California, San Diego, Feb. 2006.

[20] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "High throughput path metric for multi-hop wireless routing," in *MobiCom*. ACM, 2003.

[21] S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proc. INFOCOM*. Anchorage, AK: IEEE, May 2007.

[22] T. M. Cover, "Broadcast channels," *IEEE Trans. Inform. Theory*, vol. 18, pp. 2–14, Jan. 1972.

[23] P. P. Bergmans, "A simple converse for broadcast channels with additive white gaussian noise," *IEEE Trans. Inform. Theory*, vol. 20, pp. 279–280, Mar. 1974.

[24] R. G. Gallager, "Capacity and coding for degraded broadcast channels," *Probl. Inform. Transm.*, pp. 185–193, July 1974.

[25] Y. Wu, "Broadcasting when receivers know some messages a priori," in *Proc. Int'l Symp. Information Theory*. IEEE, June 2007.

[26] L. Xiao, T. E. Fuja, J. Kliewer, and D. J. Costello Jr., "Nested codes with multiple interpretations," in *Proc. Conf. on Inform. Sci. and Systems (CISS)*, Princeton, NJ, Mar. 2006.

[27] Z. Yang and A. Høst-Madsen, "Cooperation efficiency in the low power regime," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2005.

[28] S. Zhang, S.-C. Liew, and P. P. Lam, "Hot topic: Physical-layer network coding," in *MobiCom*. Los Angeles, CA: ACM, Sept. 2006.

[29] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2007-012, Feb. 2007.