iPack: in-Network Packet Mixing for High Throughput Wireless Mesh Networks

Richard Alimi*, Li (Erran) Li†, Ramachandran Ramjee‡, Harish Viswanathan† and Yang Richard Yang*

†Bell Labs, Alcatel-Lucent, Murray Hill, NJ, USA
‡Microsoft Research India, India
*Yale University, New Haven, CT, USA

***Abstract*—A major barrier for the adoption of wireless mesh networks is severe limits on throughput. Many in-network packet mixing techniques at the network layer [1], [2], [3] as well as the physical layer [4], [5], [6] have been shown to substantially improve throughput. However, the optimal mixing algorithm that maximizes throughput is still unknown. In this paper, we propose $iPack$, an algorithm for in-network generation of composite packets that integrates coding at two different layers of the protocol stack: XOR-based network coding and physical layer superposition coding. Using extensive simulations, we find that the throughput gain of the joint coding $iPack$ algorithm is 30% more than the better performer of network coding and superposition coding in a wide range of scenarios, and automatically takes advantage of the best available coding opportunities. In a typical wireless mesh network when more traffic is between the clients and access points, the average throughput improvement of $iPack$, our joint optimization scheduler, can be 324%, while there can be little gain (less than 10%) if network coding alone is used. We also validate our results by implementing $iPack$ on a small-scale testbed based on GNU Radio.**

I. Introduction

Wireless mesh networks are becoming a major paradigm for constructing user access networks that provide community or city-wide Internet connectivity [7]. However, recent theoretical analysis (*e.g.,* [8]) and experimental measurements (*e.g.,* [9], [10]) have shown that the current wireless mesh networks are severely limited in throughput and do not scale well as they become large and dense.

Information theoretical analysis has shown that the throughput of wireless networks can be substantially higher, if the network mixes packets for multiple flows (*e.g.,* [1], [2], [3]). A large body of previous work focuses on network-layer coding, both on theoretical bounds (*e.g.,* [11], [12], [13]), centralized algorithms (*e.g.,* [14], [15]) and distributed algorithms that require "non-local" information (*e.g.,* [16]). In-network packet mixing can also be performed at the physical layer using techniques such as superposition coding [17], [4] or physical/analog network coding [6], [5]. Superposition coding allows simultaneous transmission of different packets to multiple receivers by a *single sender* while physical/analog network coding exploits simultaneous transmission by *multiple senders* to deliver different packets to different receivers.

Given that the optimal in-network mixing algorithm that maximizes throughput is still unknown, it is not clear which set of in-network mixing techniques to use or how to combine different mixing techniques such as the physical layer and network layer coding techniques. The objective of this paper is to propose novel, simple, distributed, and easily implementable techniques for general in-network packet mixing that combines physical and network layer coding, coding schemes at two different layers of the protocol stack. The basic physical layer in-network packet mixing technique we consider is superposition coding [4] because of its implementation simplicity as compared to the difficult synchronization issues involved in multiple sender approaches such as physical/analog network coding [6], [5]. We integrate superposition coding with another simple-to-implement technique, XOR-based network coding [18], combining information from multiple flows into a composite packet or *c-packet* for short. The constructed c-packets automatically take advantage of the best available coding opportunities.

In this paper, we make the following contributions. We first formulate the *c-packet mixing problem* and show that the maximum packet mixing problem is NP-hard. Since previous work on network coding focuses only on single-rate wireless links, we propose a simple extension that allows XOR-based network coding to exploit multi-rate wireless links. We then propose two algorithms, $iGreedy$ and $iPack$, that combine coding opportunities using superposition and network coding: $iGreedy$ simply applies network and superposition coding in sequence while $iPack$ tightly integrates network and superposition coding. Based on extensive simulations, we find that that $iPack$ outperforms $iGreedy$ and achieves substantial throughput gains in multi-rate wireless mesh networks. In a wide range of scenarios, the throughput gain of $iPack$ is more than 30% of the better performer of network coding and superposition coding. There are scenarios where the gains of our solutions are quite significant. For example, in a typical wireless mesh network, when more than half of the traffic is between the clients and an access point, the average throughput gain ratio of $iPack$ is 4.24, which is 324% higher than without c-packet mixing. In contrast, there is little gain (1.05) using network coding alone. Finally, we validate our solution by implementing $iPack$ on a small-scale testbed based on GNU Radio. We find that the experimental results from the testbed are consistent with our simulations.

The rest of the paper is organized as follows. In Section II, we review related work and introduce background on basic techniques to construct c-packets. In Section III, we give a motivating example before we present in Section IV the c-packet mixing problem and analyze its complexity. Our simple, distributed $iPack$ algorithm is presented in Section V. In Section VI, using extensive evaluations, we demonstrate the throughput gains of our protocol. Our conclusions and future work are in Section VII.

II. Background and Related Work

The wireless networking community has conducted extensive research on improving wireless mesh network throughput; see [7] for a survey. However, such previous studies focus on the traditional forwarding architecture.

Several new techniques are proposed to increase mesh network throughput. One set of approaches focus on the receiver and relax the constraint that the received signal comes from a single transmitter. Instead, multiple distinct nodes may transmit simultaneously to a node, and the mixed signal is processed and/or further relayed to extract more information. Examples of such approaches include relay channels [19], [20], [21], physical network coding [6], analog network coding [5], and uplink superposition coding [20]. One potential issue with receiver-based approaches is that there can be substantial receiver complexity and synchronization requirements.

The other type of approach mixes signals and/or packets at the transmitter, and the receiver listens to a single transmitter. This approach is the focus of this paper. We leave the more general framework combining both types of approaches as future work.

There are two basic primitives for in-network packet mixing: network coding and superposition coding. Below we review the two primitives.

A. Network Coding for Packet Mixing

The network layer technique for in-network packet mixing is network coding. Due to its effectiveness, network coding has received significant attention in the research community lately. The pioneering work on network coding is by Ahlswede *et al.* [22], who showed that having the routers mix information in different messages allows the communication to achieve multicast capacity. These performance benefits have attracted many studies (*e.g.,* [23], [24], [25], [12], [13], [26], [27]). In [18], Katti *et al.* proposed COPE, the first practical network coding implementation for multiple unicast flows. Their experimental evaluations have shown substantial performance gain.

Formally, network coding takes as input K data packets $(x_1, x_2, \cdots, x_K)$ and computes a mixed packet y. Due to its simplicity and effectiveness, XOR is widely used to compute the new packet:

$$y = x_1 \oplus x_2 \oplus \cdots \oplus x_K.$$

B. Superposition Coding for Packet Mixing

A less well-studied but powerful technique for in-network packet mixing is superposition coding. The concept of superposition coding was first introduced in [17], [28], [29] by Cover *et al.* in their information theoretic study of additive white Gaussian noise (AWGN) broadcast channels. In particular, Cover showed that simultaneous transmissions to multiple receivers is more efficient than using orthogonal division of the channel. He proposed superposition coding as a technique to achieve simultaneous transmissions. In [30], Bergmans showed that this technique achieves the optimal capacity for AWGN channels. Motivated by these promising results, many researchers have since considered coding and modulation strategies (*e.g.,* [31], [32], [33], [34], [35], [36], [37]), as well as applications of superposition coding especially in the context of digital audio and video broadcasting [38]. More recently, superposition coding has been considered for standardization in cellular systems for broadcasting [39]. However, the studies on superposition coding so far have focused mainly on the physical layer.

Specifically, superposition coding is a technique by which a transmitter can *simultaneously* send independent messages to multiple receivers. In this paper, for simplicity, we restrict ourselves to the two-receiver case, in which the transmitter *superimposes* an additional message destined to a secondary receiver (receiver 2) on a basic message destined for a primary receiver (receiver 1). We also refer to the basic message as the first, primary or lower layer, and the additional message as the second, or upper layer. It is natural to extend our technique to more than two layers.

In implementation, a transmitter using superposition coding splits the available transmission power between the two layers, selects the transmission rate for each of the layers, then encodes and modulates each of the packets separately at the selected rates. The modulated symbols are scaled appropriately to match the chosen power split and mixed (summed) to obtain the final packet. For example, if quadrature phase shift keying (QPSK) is used for modulation of both layers, then the superposed transmitted signal will have a 16-point constellation.

Formally, superposition coding converts each input packet x_i to its symbol representation $S(x_i)$ and then computes a mixed packet:

$$y = a_1 S(x_1) + a_2 S(x_2) + \cdots + a_K S(x_K),$$

where a_i reflects the power allocation to the i-th input packet in the composite packet.

The two receivers decode their received signal using different schemes. Receiver 1 decodes its packet treating the superimposed additional layer as interference. Receiver 2 first decodes the basic layer, re-encodes it, and then subtracts it from the original signal. It then decodes the remaining signal. This process is referred to as *successive interference cancellation* (SIC). To allow receiver 2 to decode the basic layer whenever receiver 1 can and to ensure that the remaining signal after subtraction has enough strength over noise, the channel quality to receiver 2 should be better than that to receiver 1; that is, the two channels should be asymmetric. Thus, we also refer to receiver 1 as the weaker receiver and receiver 2 the stronger.

A major advantage of superposition coding and SIC over the traditional schemes is the expanded capacity region [40], [41], [20]. Consider a superposition c-packet with two layers. Let P be the total transmission power, and p the power allocated to the basic layer (*i.e.,* to receiver 1). Then according to the Shannon capacity formula for an AWGN channel, the achievable rate to receiver 1 is

$$\log_2 \left(1 + \frac{p|h_1|^2}{(P-p)|h_1|^2 + N_0} \right) bits/s/Hz,$$

where N_0 is the background noise. On the other hand, due to successive interference cancellation at receiver 2, the achievable rate to receiver 2 is

$$\log_2 \left(1 + \frac{(P-p)|h_2|^2}{N_0} \right) bits/s/Hz.$$

III. A Motivating Example

We first use a simple example to illustrate the benefits and basic issues of in-network packet mixing. Consider a mesh network shown in Fig. 1. There are five nodes in the network: u and v_1 to v_4. There are 4 unicast end-to-end flows in the network: $v_2 \rightarrow v_1$; $v_4 \rightarrow v_2$; $v_1 \rightarrow v_3$; and $v_3 \rightarrow v_4$. Assume all packets are of the same size sz. All 4 flows use u as an intermediate hop.

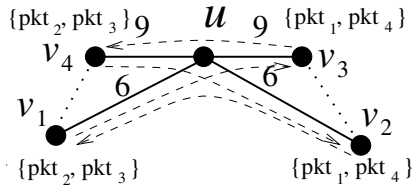

Fig. 1. A motivating example. The set at each node shows the packets available at the node.

The maximum transmission rates from u to other nodes are shown in the figure. Assume that the channels from u to v_1, v_2 are similar, at 6 Mbps; those to v_3, v_4 are similar, at 9 Mbps. Let $T = sz/6Mbps$.

Also, two nodes who are close can overhear each other's transmission: v_1 and v_4 can hear each other's transmission to u; v_2 and v_3 can hear each other's transmission to u. These overhearing links are represented by dotted links in the figure.

After the source of each flow transmits its packet to u to relay, node u has packets pkt_1, pkt_2, pkt_3, and pkt_4 for the four flows with final destinations v_1, v_2, v_3 and v_4 respectively. Each one of $v_1, \ldots, v_4$ has two packets: the one it originates and another that it overhears. The set of available packets at each one of $v_1, \ldots, v_4$ is shown in the figure.

Based on others' packet caches, u can also apply network coding and construct:

- $c_{12} = pkt_1 \oplus pkt_2$: to transmit to v_1 and v_2; the transmission rate of this coded packet should be 6 Mbps;
- $c_{13} = pkt_1 \oplus pkt_3$: to transmit to v_1 and v_3; the transmission rate of this coded packet should be 6 Mbps to accommodate the slower channel to v_1 even though the channel to v_3 can achieve 9 Mbps;
- $c_{24} = pkt_2 \oplus pkt_4$: to transmit to v_2 and v_4; the transmission rate of this coded packet should be 6 Mbps to accommodate the slower channel to v_2 even though the channel to v_4 can achieve 9 Mbps;
- $c_{34} = pkt_3 \oplus pkt_4$: to transmit to v_3 and v_4; the transmission rate of this coded packet should be 9 Mbps.

Note that we require a node to know the topology of its 2-hop neighborhood.

Now, we calculate the total transmission time for u to relay the packets. There are the following possibilities:

- No in-network packet mixing. Then, u needs 4 transmissions to relay the 4 packets to reach their destinations, with two packets at 6 Mbps and two at 9 Mbps, with a total transmission time of 10/3 T.
- Superposition coding. If u uses superposition coding, it can transmit the four packets in two transmissions, *e.g.,* one transmission of a superposition packet containing pkt_1 and pkt_3 at overall rate 6 Mbps, and another one a superposition packet containing pkt_2 and pkt_4 at overall rate 6 Mbps. The total time is 2 T.
- Network coding. If u uses network coding, then it can forward the packets to their final destinations in two transmissions (c_{12} and c_{34}) with a total time of 5/3 T.
- Integrated superposition coding and network coding (case 1). Node u can further increase throughput by combining network coded packets using superposition coding. If u mixes the two network coded packets c_{12} and c_{34} using superposition coding, it can transmit both using a single transmission with time duration T.
- Integrated superposition coding and network coding (case 2). If u were to choose c_{13}, then it could not combine c_{13} with c_{24} using superposition coding due to the lower channel gain to v_2. Thus, it could mix c_{13} only with pkt_4 using superposition coding. Then it would need an additional transmission for pkt_2 to v_2 at 6 Mbps. The total time will be 2 T.

IV. Problem Formulation

We now formalize the distributed in-network packet mixing problem. Since we are interested in distributed algorithms, we consider a specific node u. Fig. 2 illustrates the setup.

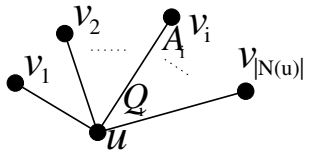

Fig. 2. Problem formulation.

Let $N(u)$ denote the neighbor set of u. The set of packets to be transmitted from u to its neighbors is Q. Node u queues the packets in Q to the output interface queues, one for each neighbor. Let Q_i denote the output interface queue to neighbor v_i. Let $pkt_{i,f}$ be the head-of-line packet at Q_i for flow f. Node u also maintains a cache of packets it has transmitted or received through opportunistic listening. Let this set of packets be C. Such packets can be used by u to construct packets to send to its neighbors or by the neighbors to construct packets to send to u.

Node u also maintains state about its neighbors. Assume that the channel gain from u to each neighbor $v_i \in N(u)$ is measured. Also, u estimates the packet availability at its neighbors. Let A_i denote the set of packets u knows are available at neighbor v_i.

For implementation simplicity, we use two-layer superposition coding. Given two packets pkt_1 to v_1 and pkt_2 to v_2, we can determine whether they can be transmitted together using superposition coding and at which data rate. We allow an unlimited number of packets in XOR-based network coding construction.

Given this setup, we would like to implement in-network packet mixing such that maximal number of packets in Q will be included (assuming reliable transmission), disallowing out-of-order packets for each flow. We refer to this problem as the maximum packet mixing problem. Unfortunately, this problem is NP-hard.

Theorem 1: The maximum packet mixing problem is NP-hard.

mnetcode(Q, p)
01. **foreach** discrete rate r_i
02. (S_i, V_i)=snetcode(Q, r_i, p);
03. **endfor**
04. $i^* = argmax_i |S_i| \times r_i$
05. $pkt^* = XOR(S_{i^*})$
06. return (pkt^*, V_{i^*}, S_{i^*} and r_{i^*})

Fig. 3. Multi-rate opportunistic network coding at node u.

Proof: We only sketch the proof. The reduction is from the maximum clique problem. Given a graph $G = (V, E)$ in the maximum clique problem, we construct an instance of the maximum packet problem. We first create a node u. For each node $w_i \in V$, we create a packet pkt_i and a node v_i such that pkt_i is destined to next hop v_i. For each edge $(w_i, w_j) \in E$, we assume $pkt_i \in A_{v_j}$, and $pkt_j \in A_{v_i}$. We create pkt_0 that is destined to v_0. pkt_0 is in $A_{v_i}, \forall i \neq 0$. Let the index of each packet be its position in the output queue of node u. pkt_0 is the head of line packet. It is easy to see that we can have a packet of size k iff $G = (V, E)$ has a clique of size $k - 1$. ■

V. Protocol Design

Although the hardness result from the preceding section shows that the full gains of in-network packet mixing may be computationally hard to achieve, we can still design approximation algorithms to improve wireless network throughput. As we will show, there are simple, distributed protocols to substantially improve network capacity in practical settings.

We present our design in several steps: we first extend network coding to multi-rate networks, as this not only improves network coding efficiency but also is necessary to integrate with superposition coding; then we design a greedy algorithm for in-network packet mixing combining superposition coding and network coding; finally we present our joint optimization algorithm iPack.

A. Opportunistic Multi-rate Network Coding

Since the previous work focuses on single-rate network coding but many wireless mesh networks allow multi-rate adaptation, we first present a simple extension to generate network coded packets in multi-rate mesh networks. The pseudo-code is shown in Fig. 3.

Recall that Q is the data structure containing the packets to be transmitted from u. Let p be the maximum transmission power allowed to transmit the mixed packet. Since the network allows multiple discrete rates, the algorithm iterates over all rates and selects the best one.

Specifically, for each discrete rate r_i, the algorithm invokes a single-rate network coding algorithm, $snetcode$, to maximize the number of packets coded. The procedure $mnetcode$ can be based on any standard single-rate network coding algorithm (*e.g.,* [18]) with two extensions. The first extension is that since there may be only a subset of the neighbors that can successfully decode packets sent with that rate, $snetcode$ should first prune neighbors. The second extension is to integrate with superposition coding. Specifically, since each link has a different data rate, we need to use the minimal rate among all links that need to receive the coded packet. We try to code packets in decreasing order of link rates. The idea is that packets with similar channel conditions should be coded together. This will give superposition coding more freedom in choosing which packets to code and at which data rate. If multiple links have the same data rate, we randomize the order so that each link will get a fair chance of being selected.

The result of $snetcode$ at each rate r_i consists of two items: S_i, the set of packets that should be XOR-ed together; and V_i, the set of receivers. Since the throughput of each such coded packet depends on both the number of packets coded and the rate at which we can transmit the packet, we compute the effective throughput as the product of the cardinality of S_i and the rate at which the coded packet is transmitted. We pick the discrete rate r_{i^*} that gives the maximum throughput. The coded packet is the XOR of the packets in S_{i^*}.

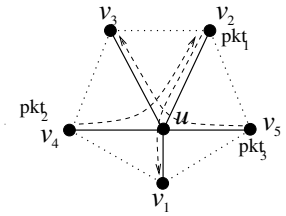

Fig. 4. An example to illustrate $mnetcode$. Packets are shown at the original sources.

Now we give a simple example shown in Fig. 4 to illustrate the basic idea of the algorithm. We assume all packets have the same size. We ignore header overhead for simplicity. The network has 3 end-to-end packets pkt_1, pkt_2, pkt_3 to be transmitted from v_2, v_4, v_5 to their final destinations v_1, v_2, v_3 respectively. Node u is the node relaying traffic for the other nodes. The paths of the end-to-end packets are labeled with dashed lines. The overhearing links are labeled with dotted links. For example, v_4 has two dotted links to v_1, v_3. Thus, when it transmits to u, v_1, v_3 can overhear the transmission but v_2, v_5 cannot.

After sources v_2, v_4, v_5 transmit, v_1, v_2, v_3 will each have two of the three packets pkt_1, pkt_2, pkt_3 through either overhearing or because it sent the packet. Assume the rate of link (u, v_1) is $2r$ while the rate of other links is r.

Now u iterates over the two rates $2r$ and r. At rate $2r$, it can send pkt_1 only, and the cardinality and rate product is $2r$. At rate r, it can code all three packets together and achieves a cardinality and rate product of $3r = 3 \times r$. Thus, u transmits at rate r.

A property of $mnetcode$ is that it does not guarantee that the throughput of each flow will be better than its throughput without using network coding. This may not be obvious, but can be illustrated by an example. Assume that node u transmits to three nodes, v_1, v_2, v_3. Assume that the queues to v_1, v_2, v_3 are always backlogged. Suppose one can always send XOR-ed packets to v_1, v_2, and to v_2, v_3. Let the channel rates to v_1, v_2, v_3 be $2, 1, 2$, respectively. Without network coding, and if u implements equal time sharing among the three queues, then the throughput to the three receivers will be $2/3, 1/3, 2/3$ respectively. With network coding ($mnetcode$), the throughput will become $1/2, 1, 1/2$.

If a design policy is that the throughput of the flows should only improve if network coding is used, we can use a simple, novel technique to achieve the goal. In particular, we can resolve the issue by introducing a finite-valued lookahead factor D where each packet is allowed to code only with packets

iGreedy(Q)
01. (pkt, V, S, r) = mnetcode(Q, P)
02. $Q = Q - S$
03. enqueHead(Q, pkt)
04. call supercode(Q)

Fig. 5. Algorithm iGreedy at node u.

within D distance away. The distance is in terms of arrival time into the queue in discrete units.

Theorem 2: Given a scheduler that schedules unicast packets in a certain order at node u without network coding. Assume that the $mnetcode$ procedure respects that order. In addition, a packet does not code with packets more than D away. Assume channel rate does not vary, and the number of coding opportunities grow unbounded over time. Let the long-term throughput achieved by the link (u, v_i) be $tput_i$ and $tput'_i$ for without and with network coding respectively. We claim that $tput'_i > tput_i$.

Proof: We only sketch the proof. Let the total transmission times without and with network coding be $tx(n)$ and $tx'(n)$, respectively, when the n-th packet is being transmitted. Given that each coded packet reduces the overall transmission time when compared with no coding case, the total transmission time difference $tx(n) - tx'(n)$ grows unbounded assuming infinite coding opportunities. At time n, there can be only a finite D number of future packets that get scheduled in the case of network coding. Let the time to transmit these packets be T_D. Then, $tput' = n/(tx'(n) - T_D)$ and $tput = n/tx(n)$. We have that $tput' > tput$ after a certain n. ■

B. Packet Mixing Algorithms

We now describe two packet mixing algorithms, $iGreedy$ and $iPack$ that combine network and physical-layer coding. Since a network may already have a MAC scheduler (*e.g.,* FIFO, round robin, or proportional fairness) to determine which data packet to transmit next, both algorithms assume that this scheduler has already picked a head-of-line packet pkt_{hol}. Both algorithms invoke the multi-rate network coding algorithm from the preceding section. The $iGreedy$ algorithm applies network and superposition coding in sequence. The $iPack$ algorithm integrates network and superposition coding by maximizing the total transmission rate under the constraint of transmitting the head of line packet, or more generally for some fairness criterion.

$iGreedy$: Greedy Packet Mixing

With the preceding $mnetcode$, it is straightforward to design a greedy algorithm for in-network packet mixing that incorporates both network coding and superposition coding. The algorithm, named $iGreedy$, is shown in Fig. 5.

In this algorithm, we first try to find the coded packet pkt^* using procedure $mnetcode$ defined in Fig. 3. This packet will be used for the first layer in the SC coded packet. We then call a superposition coding procedure on the new queue $Q - S$. Similar to $mnetcode$, we need to modify the $supercode$ procedure to implement pruning of links with inadequate channel support. We use the $Gopp$ algorithm [4] as the superposition coding procedure in this paper, which selects superposition coded packets that take advantage of the discrete rates currently supported by the receivers.

iPack(Q) — pkt_{hol} is the packet to schedule next.
01. $T^* = 0$ // Best observed throughput
02. **for** each discrete rate r_i that pkt_{hol} supports
03. // Determine first layer packet and power level; snetcode
04. // must select pkt_{hol}
05. $Q' = Q$
06. (S_i, V_i) = snetcode(Q', r_i, P)
07. determine min power p_1 s.t. receivers in V_i support r_i
08. // Find best rate and packets for second layer
09. (n_2, pkt_2, r_2) = bestSecondLayer($Q' - S_1$, r_i, p_1)
10. // Update best observed throughput if necessary
11. **if** $(n_2 + |S_i|) \times r_i > T^*$ **then**
12. $T^* = (n_2 + |S_i|) \times r_i$
13. ${pkt_1}^* = XOR(S_i)$
14. ${pkt_2}^* = pkt_2$
15. ${r_1}^* = r_i$, ${p_1}^* = p_1$, ${r_2}^* = r_2$
16. **endif**
17. **endfor**
18. // Compute throughput without superposition coding
19. $({pkt_m}^*, V_m, S_m, r_m)$ = mnetcode(Q, P)
20. // Transmit with scheme that provides best throughput
21. **if** $T^* > |S_m| \times r_m$ **then**
22. transmit encode(${r_1}^*$, ${p_1}^*$, ${pkt_1}^*$, ${r_2}^*$, ${pkt_2}^*$)
23. **else**
24. transmit ${pkt_m}^*$
25. **endif**

Fig. 6. Algorithm iPack at node u.

bestSecondLayer(Q, r_1, p_1) – r_1 is rate of the first layer; p_1 is power of the first layer.
01. $n^* = 0$; $pkt^* = \emptyset$; $r^* = 0$
02. **foreach** second-layer discrete rate r
03. $Q' = Q$
04. $pkt = \emptyset$ // set of network-coded packets
05. $n = 0$ // number of raw packets can be packed
06. **repeat** to select $\lfloor \frac{r}{r_1} \rfloor$ network-coded packets
07. (S, V) = snetcode(Q', r, $P - p_1$)
08. $n = n + |S|$
09. $pkt = pkt \cup \{XOR(S)\}$
10. $Q' = Q' - S$
11. **endrepeat**
12. **if** $n > n^*$ **then**
13. $n^* = n$, $pkt^* = pkt$, $r^* = r$
14. **endif**
15. **endfor**
16. **return** (n^*, pkt^*, r^*)

Fig. 7. Algorithm to select second layer packets by packing the largest number of raw packets.

$iPack$: Integrated Packet Mixing

A key advantage of $iGreedy$ is its simplicity. However, it leaves some coding opportunities unexplored. An example can be seen in Section III. We now present $iPack$, an algorithm that jointly optimizes the use of superposition and network coding. The algorithm is shown in Fig. 6.

$iPack$ computes rates for the first and second layers of superposition coding with each layer composed of a set of appropriate network coded packets. Specifically, $iPack$ iterates over each rate r_i that the head-of-line packet can support, finding the network coded packet $XOR(S_i)$ for the first layer (line 6) and computing the minimum power p_1 required for it to be received by all of its intended receivers V_i (line 7), assuming $P - p_1$ is treated as noise.

For the remaining power $P - p_1$, the algorithm calls $bestSecondLayer$ to find the rate r_2 which can pack the max-

imum number of packets in the second layer. The pseudocode for $bestSecondLayer$ can be found in Fig. 7. The result of $bestSecondLayer$ consists of three items: n, the number of raw packets packed; pkt, the set of network coded packets to encode the n raw packets; and r, the rate of layer 2 (line 9). Given the coded packets for the two layers, the algorithm calculates the total number of packets transmitted, normalizes according to the rate of the first layer, and selects the superposition coded packets with the highest rate (line 11-16). Finally, the algorithm compares the throughput using superposition coding and that without superposition coding using $mnetcode$ and selects the one with the higher throughput (line 19-25).

VI. Evaluations

We use ns-2 (Ver. 2.31) as our simulation evaluation environment. We have also conducted experimental evaluations on a small-scale testbed implemented using GNU Radio.

A. Simulation Methodology

Physical-layer encoding and decoding
For our simulation setup, we use the 2-ray ground model with 0 dBi antenna gains and antenna heights of 1 meter. To implement a realistic packet decoding model, we use a lookup table for packet error rates (PER) [42] given an observed SNR. There is a separate PER curve for each 802.11g OFDM data rate. The SNR depends on the received signal power P_r, noise power $N_0 = -86$ dBm, and implementation margin $I_m = 5$ dB specified in the 802.11g standard. These parameters produce maximum transmission ranges similar to those of the Cisco Aironet 802.11g in a typical outdoor environment [43].

When a packet is received in the physical layer, we decode it as follows. If the packet is not a superposition c-packet, we compute $\mathrm{SNR} = \frac{P_r I_m}{N_0}$ and look up the PER corresponding to the data rate at which the packet was transmitted. The packet is dropped with a probability equal to the PER.

If the packet is a superposition c-packet, we compute the SNR for the basic layer as $\mathrm{SNR} = \frac{P_r p_1 I_m}{P_r(1-p_1)+N_0}$ where $p_1 \in [0,1]$ is the fraction of power allocated to the basic layer packet. The SNR lookup and decoding for the basic layer are done in the same way as a non-superposition c-packet. If the basic layer is decoded successfully, we compute the SNR for the additional layer as $\mathrm{SNR} = \frac{P_r(1-p_1)I_m}{N_0}$. SNR lookup and decoding for the additional layer are also done in the same way as the non-superposition c-packet.

To handle channel estimation, the physical layer calculates the channel gain $h = \frac{P_r}{P}$ for each received transmission. This value along with the transmission source address is passed up to the MAC layer, which manages a table of channel gains for each link. These channel gains are used to compute SNR values in the scheduling and routing layers.

Link and network layer and processing
We assume that all transmitted packets are 1500 bytes. We also disable RTS/CTS.

Note that there may be multiple packets packed into the additional layer of a superposition coded packet. These are unpacked in the MAC layer before being delivered further up the protocol stack.

Network coding requires nodes to operate in promiscuous mode and buffer overheard packets. Each overheard packet times out after 1 second and is discarded. Decoding of c-packets using network coding is implemented at the network layer. The packet addressed to the node is passed to the upper layers only if all other packets that had been coded together had been previously overheard.

Topology, traffic demands and routing
We deploy the clients and access points (APs) in a 500 meter by 500 meter region. When there is a single access point, it is located in the center. Clients are distributed uniformly random within the square. Note that non-uniform distribution of nodes can increase channel diversity and thus may improve the effectives of superposition coding based in-network packet mixing. Access points always have their total transmit power set to 200 mW (maximum transmission range 335 m). Unless otherwise specified, all clients have total transmit powers set to 100 mW (maximum transmission range 281 m) and all traffic flows are between client nodes relayed through the access points. When there are multiple access points, they form a wireless mesh network to relay traffic. Then, a static shortest path routing is computed with link metrics inversely proportional to the maximum supported data rate along the link. A client associates with the access point with the best channel and communicates through that access point.

Evaluated protocols
We evaluate the following c-packet construction protocols.

- $Gopp$: the c-packet construction protocol based only on superposition coding [4];
- $mnetcode$: presented in Fig. 3, which is an extension of the XOR-based network coding technique to handle multi-rate network coding (COPE [18] is used for single-rate network coding);
- $iGreedy$: presented in Fig. 5;
- $iPack$: presented in Fig. 6.

Performance metrics
We use the FIFO scheduler with no c-packets as the base case. Our primary metric is throughput gain ratio (or throughput gain for short) which is defined as the aggregate throughput achieved by any of the four c-packet construction protocols divided by the throughput achieved by the FIFO scheduler. We also discuss loss rate and fairness.

B. Simulation Results

A single access point
We start our evaluations with a simple scenario: a single access point in the network. We place 10 clients uniformly random in the network and setup 8 random flows between the clients. We vary the UDP traffic rate of the flows. This changes the traffic demand to the network. It is intuitive that the main advantage of the c-packets is that they extend the capacity region of the network.

Fig. 8 confirms the intuition. In this plot, each point represents the result of one particular instance of an experiment. Each experiment is performed with 30 randomly-generated topologies. We also show the average highlighted by a bold horizontal line. From the figure, we observe that when the traffic demand is low, the gain of c-packets is low. This can be explained since the algorithms depend on the availability of

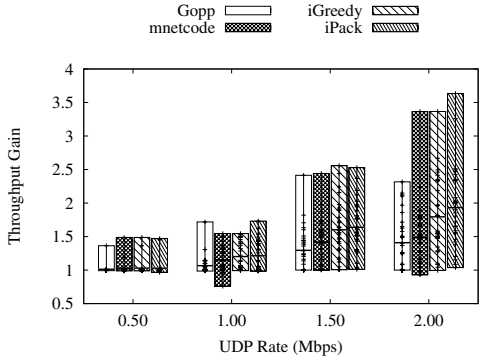

Fig. 8. Throughput gains with varying flow rate.

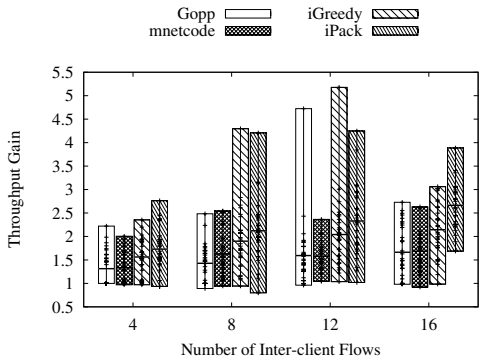

Fig. 9. Throughput gains with increasing numbers of flows between clients.

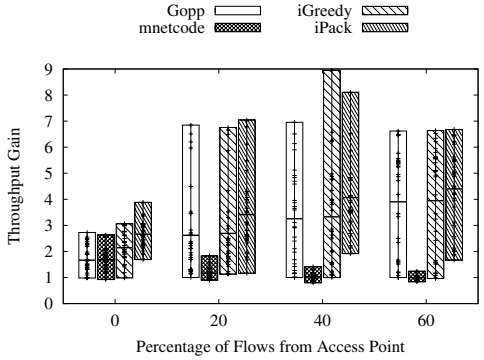

Fig. 10. Throughput gains with increasing flow from the access point.

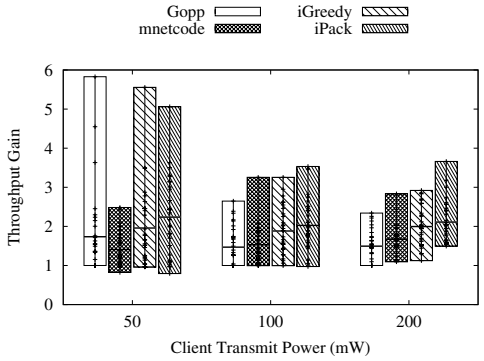

Fig. 11. Throughput gains with varying client transmit powers.

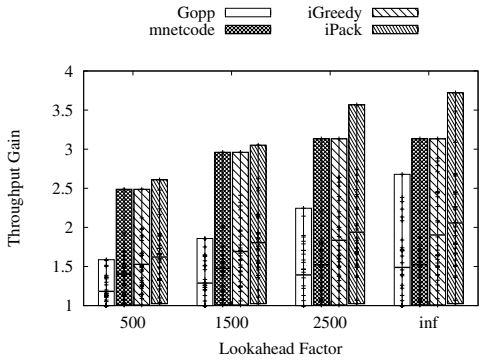

Fig. 12. Throughput gains with increasing lookahead factor.

packets in the queues for constructing c-packets. However, since the traffic demand is low, the network can handle the demand well and thus the queues are mostly empty, presenting low opportunities for c-packet construction. As demand increases, without c-packets, the queues grow and the network approaches congestion collapse. With c-packets, however, the packets in the queues present construction opportunities and the gain increases. For example, for $Gopp$ and $mnetcode$, the average gain increases from slightly above 1 at low demand to above 1.4 at high demand.

However, the four algorithms achieve different levels of gains. From Fig. 8, we observe that the gains of general c-packet construction is consistently higher than using superposition coding or network coding alone. At per-flow demand rate 2 Mbps, the gain of $iPack$ is 30% and 38% higher than $Gopp$ and $mnetcode$, respectively. We observe the maximum throughput gain of $iPack$ to be 3.6, representing a 2.6 times throughput increase over the case with no c-packets.

Another way to vary network traffic demand is to vary the number of flows. We use 1 access point and 20 clients, but vary the number of flows between the clients. All flows are infinitely backlogged. Fig. 9 shows the result. For $Gopp$ and $mnetcode$, the average gain increases from 1.31 and 1.33 to 1.66 and 1.68, respectively. For $iGreedy$ and $iPack$, the average gain increases from 1.56 and 1.73 to 2.14 and 2.66, respectively. The gain of $iPack$ is 58% more than $Gopp$ and $mnetcode$, and is 24% more than $iGreedy$.

In a typical wireless mesh network, most of the traffic demand might be between the clients and the access points, which are connected to the Internet, rather than between clients. Fig. 10 shows the results for this setting. The network has 1 access point and 20 clients. There are 16 flows in each scenario, but we vary the percentage of the flows with the source at the access point. If there is no client to client traffic, then there are no network coding opportunities. Thus, as the percentage of flows from the access point increases, $mnetcode$'s average throughput gain drops from 1.68 to almost 1. On the other hand, the opportunities for superposition coding increase as the percentage of flows from the access point increases. $Gopp$'s average throughput gain increases from 1.66 to 3.71. $iPack$'s average throughput gain increases from 2.66 to 4.24, which is an increase of 324%. This result demonstrates that our solutions are able to exploit both network coding and superposition coding opportunities effectively.

Transmission power may have impacts on the loss rate and the probability of overhearing. Fig. 11 evaluates the algorithms as we vary the transmission power of the clients. The network has 1 access point, 10 clients, and 8 flows. We observe that at a low transmission power, the gain spreads over a wider range than that at a high transmission power. However, the average gains are relatively stable.

It is also important to evaluate the performance of individual flows. As shown before, the lookahead factor has an impact on per-flow performance. Fig. 12 plots the throughput gain with varying values of D. There is no lookahead constraint imposed on the scenarios with $D = \infty$. The network has 1 access point, 10 clients, and 8 flows. The plot shows that there are additional throughput gains as D increases since more coding opportunities are available.

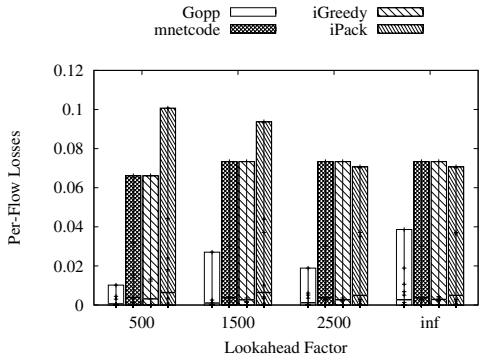

Fig. 13. Per-flow losses with increasing lookahead factor.

Fig. 13 shows the effect on the throughput of individual flows. Per-flow losses are calculated as the lost throughput on individual flows summed over the flows that performed worse than under the FIFO scheduler. The resulting lost throughput is then normalized by the aggregate throughput of the FIFO scheduler. The plot shows that even as D increases, there is no noticeable impact on individual flows. We have also evaluated the impact on per-flow throughput using the fairness index proposed by Jain, Chiu, and Hawe [44]. We observe similar results.

Mesh network of access points

We next evaluate the performance as we increase the size of the access-point mesh network. Fig. 14 plots the aggregate throughput achieved by each of the schedulers as we increase the size of access-point mesh network. When there are 2 access points, they are located at $(250, 125)$ and $(250, 375)$ and there are 20 clients and 16 flows. When there are 4 access points, they are located at $(125, 125)$, $(125, 375)$, $(375, 125)$, and $(375, 375)$ and there are 40 clients and 32 flows. We observe that, as the size of the access-point network increases, the amount of opportunities for superposition coding and network coding increases. Therefore, all four algorithms show a throughput increase. For *Gopp* and $mnetcode$, the average gain increases from 1.47 and 1.53 to 1.88 and 1.81, respectively. For $iGreedy$ and $iPack$, the average gain increases from 1.88 and 2.02 to 2.36 and 2.60, respectively. $iPack$'s throughput is more than 30% more than *Gopp* and $mnetcode$ in all cases.

Finally, we evaluate the performance on a larger mesh network with a subset of the access points acting as gateways to the Internet, which is typical of deployed mesh networks. Fig. 15 plots the aggregate throughput of for networks over an area with 12 access points, 5 of which are randomly-selected as gateways. There are 50 clients and 40 flows, and we vary the percentage of flows from gateways to clients. The remainder of the flows are between clients within the mesh network. The results show that the schedulers are able to provide significant gains even as the size of the network increases. In particular, the average gain

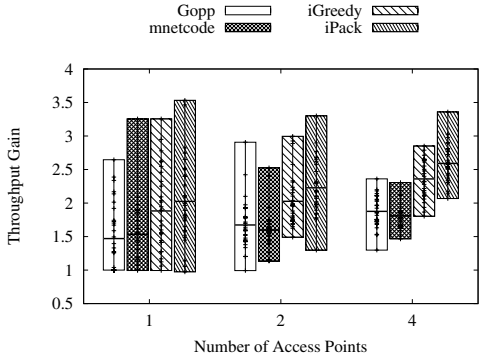

Fig. 14. Throughput gains of mesh networks with increasing access-point mesh network.

for $iPack$ is as high as 2.65, while $iGreedy$ achieves average gains as high as 2.24. One possible explanation for the limited gains of $mnetcode$ is that there is a limited amount of traffic amongst the clients of a particular access point.

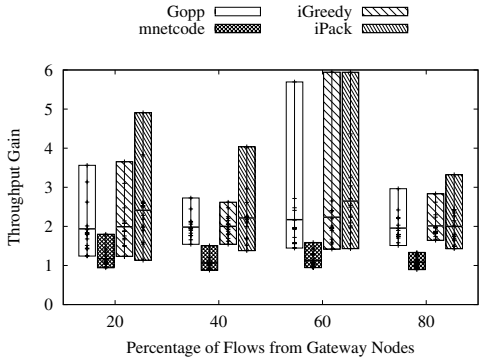

Fig. 15. Throughput gains of mesh networks with 12 access points, 5 of which are gateways.

C. Testbed Results

In addition to simulation, we also implement the coding schemes in a small-scale testbed using GNU Radio [45] to observe their performance on real wireless links. Our implementation consists of about 4000 lines of C++ code and about 3000 lines of Python code and includes an implementation of the DCF mode of the 802.11 MAC. Different from the simulations which transmit UDP packets with no link layer reliability, the testbed implements link layer reliability so that we can measure the expected transmission time for each coding scheme. For simplicity, we consider persistent retransmission; that is, we repeat the same c-packets until each receiver is able to recover its packet. This may not be necessary as the transmitter can adapt the coding scheme after some receivers have received their packets.

Our testbed evaluation uses the network shown in Fig. 1, with 5 nodes and 4 unicast flows. Nodes $v_1, \ldots, v_4$ send normal data packets, while u incorporates packet mixing. For superposition coding, u allocates 70% of its power to the basic layer and the remaining 30% to the additional layer.

We report in Table I the mean normalized total transmission time for all four receivers to recover their packets, where one unit of time is the time to transmit one packet at 6 Mbps. The throughput gains are largely consistent with our preceding simulation results.

Scheme	Norm. exp. trans. time	Gain ratio
No Coding	3.92	1
Superposition	2.88	1.4
Network coding	2.30	1.7
iPack	2.07	2.0

TABLE I
TESTBED PERFORMANCE FOR THE NETWORK SHOWN IN FIG. 1.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have designed a new wireless mesh network algorithm based on in-network packet mixing. We demonstrated that our practical and simple protocols are highly effective in improving wireless mesh network throughput. It consistently outperforms the better performer of network coding and superposition coding. We also validate our solutions on a small-scale testbed implemented using GNU Radio.

There are many avenues for future studies. First, we would like to build a large-scale network consisting of prototype nodes to evaluate the performance of our system. Such evaluations may reveal further opportunities to improve network throughput. Second, the focus of this paper is on in-network packet mixing at the transmitter. It is possible to mix packets/signals during the receiving process. Integrated design may further improve performance. Third, we use single-hop packet mixing. That is, a composite packet can be decoded in one-hop. It is possible to allow multi-hop packet mixing. Multihop network coding has already been well studied. Evaluating the performance of multihop network coding integrated with superposition coding is an interesting topic.

REFERENCES

[1] K. Jain, V. V. Vazirani, and G. Yuval, "On the capacity of multiple unicast sessions in undirected graphs," *IEEE/ACM Trans. on Networking*, vol. 14, 2006.

[2] Z. Li and B. Li, "Network coding: The case of multiple unicast sessions," in *Proc. of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2006.

[3] L.-L. Xie and P. Kumar, "Wireless network information theory," *IEEE/ACM Trans. on Networking*, 2006.

[4] L. E. Li and *et al.*, "Extended abstract: Superposition coding for wireless mesh networks," in *Proc. of MobiCom*, Sep. 2007.

[5] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *Proc. of ACM SIGCOMM '07*, Kyoto, Japan, Aug. 2007.

[6] S. Zhang, S.-C. Liew, and P. P. Lam, "Hot topic: Physical-layer network coding," in *Proc. of MobiCom*, Sep. 2006.

[7] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, no. 9, 2005.

[8] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 388–404, Jan. 2001.

[9] "Ugly truth about mesh networks," http://www.dailywireless.org/2004/06/28/ugly-truth-about-mesh-networks/.

[10] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. of MobiCom*, Rome, Italy, Jul. 2001.

[11] N. Harvey, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *IEEE/ACM Trans. on Networking*, vol. 52, no. 6, 2006.

[12] J. Liu, D. Goeckel, and D. Towsley, "Bounds on the gain of network coding and broadcasting in wireless networks," in *Proc. of IEEE INFOCOM '07*, Anchorage, AK, May 2007.

[13] S. Song, D. Goeckel, and D. Towsley, "Collaboration improves the connectivity of wireless networks," in *Proc. of IEEE INFOCOM '06*, Apr. 2006.

[14] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Medard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Prof. of ISIT*, 2006.

[15] Y. Wu, P. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. on Communications*, 2005.

[16] D. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. of IEEE INFOCOM*, Mar. 2005.

[17] T. M. Cover, "Broadcast channels," *IEEE Trans. on IT*, vol. IT-18, pp. 2–14, 1972.

[18] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proc. of ACM SIGCOMM '06*, Pisa, Italy, Sep. 2006.

[19] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *IEEE Trans. on IT*, Feb. 2004.

[20] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, May 2005.

[21] Y. E. Liu, "A low complexity protocol for relay channels employing rateless codes and acknowledgement," in *Proc. of ISIT*, Seattle, WA, Jul. 2006.

[22] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. on IT*, 2000.

[23] S. Bhadra, P. Gupta, and S. Shakkottai, "On network coding for interference networks," in *Proc. of ISIT*, Seattle, WA, Jul. 2006.

[24] T. Ho, R. Koetter, M. Medard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. on IT*, vol. 52, no. 10, 2006.

[25] J. Jin, T. Ho, and H. Viswanathan, "Comparison of network coding and non-network coding schemes for multi-hop wireless networks," in *Proc. of ISIT*, Seattle, WA, Jul. 2006.

[26] S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proc. of IEEE INFOCOM '07*, Anchorage, AK, May 2007.

[27] C. Wang and N. B. Shroff, "Beyond the butterfly - a graph-theoretic characterization of the feasibility of network coding with two simple unicast sessions," in *Proc. of ISIT*, Nice, France, Jun. 2007.

[28] P. P. Bergmans and T. M. Cover, "Cooperative broadcasting," *IEEE Trans. on IT*, vol. IT-20, pp. 317–324, 1974.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons,, 1991.

[30] P. P. Bergmans, "A simple converse for broadcast channels with additive white Gaussian noise," *IEEE Trans. on IT*, vol. IT-20, pp. 279–280, 1974.

[31] M. Airy, A. Forenza, R. W. Heath, and S. Shakkottai, "Practical Costa precoding for the multiple antenna broadcast channel," in *Proc. of GLOBECOM*, vol. 6, 2004.

[32] A. R. Calderbank and N. Seshadri, "Multilevel codes for unequal error protection," *IEEE Trans. on IT*, vol. 39, no. 4, pp. 1234–1248, 1993.

[33] M. B. Pursley and J. M. Shea, "Multimedia multicast wireless communications with phase-shift-key modulation and convolutional coding," *IEEE Journal of Selected Areas in Communications*, vol. 17, Nov. 1999.

[34] ——, "Nonuniform phase-shift-key modulation for multimedia multicast transmission in mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, vol. 5, May 1999.

[35] S. Shamai, "A broadcast approach for the multiple-access slow fading channel," in *Proc. of ISIT*, 2000.

[36] S. Vishwanath, N. Jindal, and A. Goldsmith, "Duality, achievable rates, and sum-rate capacity of Gaussian MIMO broadcast channels," *IEEE Trans. on IT*, vol. 49, no. 10, 2003.

[37] W. Yu and J. Cioffi, "Trellis precoding for the broadcast channel," in *Proc. of GLOBECOM*, vol. 2, 2001.

[38] K. Ramachandran, A. Ortega, K. M. Uz, and M. Vetterli, "Multiresolution broadcast for digital HDTV using joint source channel coding," *IEEE Journal of Selected Areas in Communications*, vol. 11, pp. 6–23, 1993.

[39] 3rd Generation Partnership Project 2, "Physical layer for ultra mobilebroadband (UMB) air interface specification," 3GPP2 C.P0084-001, Available from www.3gpp2.org, Feb. 2007.

[40] P. Gupta, "Towards an information theory of large networks: An achievable rate region," *IEEE Trans. on IT*, vol. 49, no. 8, Aug. 2003.

[41] S. Toumpis and A. Goldsmith, "Capacity regions for wireless ad hoc networks," *IEEE Trans. on Wireless Comm.*, vol. 2, no. 4, Jul. 2003.

[42] K.-Y. Doo, J. young Song, and D.-H. Cho, "Enhanced transmission mode selection in IEEE 802.11a WLAN system," in *Proc. of Vehicular Technology Conference*, Sep. 2004, pp. 5059–5062.

[43] "Cisco Aironet 802.11a/b/g," Available from http://www.cisco.com/application/pdf/en/us/guest/products/ps5818/c1650/ccmigration_09186a00801ebc29.pdf.

[44] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC, Tech. Rep. TR-301, 1984.

[45] "GNU Radio: The GNU Software Radio," http://www.gnu.org/software/gnuradio/.