NCOM: Network Coding based Overlay Multicast in Wireless Networks

Tan Le, Xing Chen and Yong Liu
Department of Electrical and Computer Engineering
Polytechnic Institute of New York University
Brooklyn, New York, 11201, USA
Email: tle04@students.poly.edu, yongliu@poly.edu

Abstract

The capacity of wireless networks are increasingly challenged by the traffic stresses generated by data-intensive applications. Multicast is a bandwidth-efficient solution to simultaneously disseminate data to multiple receivers. In this paper, we present NCOM, a network coding based overlay multicast design, that integrates Network Coding (NC), Opportunistic Routing (OR), and cross-layer link scheduling to achieve high efficiency and reliability multi-hop wireless multicast. In NCOM, the source and receivers are connected by an overlay Steiner tree optimized for the minimum OR distance between nodes. With NC, coded packets are opportunistically transmitted along overlay links. The transmissions of adjacent nodes in the overlay multicast are coordinated by a novel multicast acknowledgement and cross-layer MAC scheduling. We implement NCOM in OPNET by customizing the IEEE 802.11b modules. Through OPNET simulations, we demonstrate that NCOM can achieve a higher throughput and lower source transmission redundancy than the existing NC and OR based wireless multicast designs. We also analyze the advantages of NCOM over Pacifier in difference case study. NCOM can be easily deployed for efficient and reliable multicast in multi-hop wireless networks.

I. Introduction

The increasing popularity of wireless devices and new wireless applications makes it important to deliver multicast services efficiently over multi-hop wireless networks. It is well-known that the general minimum-cost multicast routing problem is NP-hard. Wireless multicast additionally has to deal with lossy packet transmissions on volatile wireless links. Recent research advances Network Coding (NC), Opportunistic Routing (OR), and optimal cross-layer scheduling present new opportunities to achieve high efficiency in wireless multicast. With NC [1], packets are mixed at the source node as well as relay nodes. A receiver can decode the original packets upon receiving enough number of independent coded packets. NC has been adopted to improve the efficiency of multicast in wireline networks [2] and multihop wireless networks [3], [4], [5]. OR [6] exploits the broadcast nature of wireless transmission and opportunistic packet receptions to significantly reduce the number of transmissions necessary to

deliver a packet. It was theoretically shown that the network capacity can be achieved through cross-layer operations involving source rate control, network routing, and link scheduling, for both unicast flows [7], [8], [9], [10], [11] and multicast flows [12], [13], [14].

Some limited efforts have been made to integrate NC and OR in wireless multicast. In MORE [3], a source node firstly calculates for each receiver the OR forwarder set and the expected number of transmissions of each forwarder based on the link ETX metric [15]. Then the forwarder sets for multiple receivers are merged and the transmission credits of each forwarder are updated. However, the merge of forwarder sets for different receivers is not necessarily efficient. As a result, MORE incurs high data transmission redundancy on source and relay nodes in multicast. A more recent work of adopting NC and OR in wireless multicast is Pacifier [4]. The source first calculates the shortest path tree to reach all receivers based on the link ETX metric. To exploit the OR gain, a node not only receives packets from its ancestor nodes, but also can overhear packets from its sibling nodes. In Pacifier, the construction of multicast tree does not explicitly take into account the opportunistic packet reception between sibling nodes. The constructed tree is therefore sub-optimal under OR. In their experiments, Pacifier increases the average throughput over MORE by 171%. However, it still suffers very high source redundancy. Their experiments showed that for Pacifier, the source transmits on average 5.84 times the original data size while in MORE the source transmits on average 17 times the data size.

In this paper, we adopt a different approach, Network Coding based Overlay Multicast (NCOM), that efficiently integrates NC, OR and cross-layer scheduling in wireless multicast. In NCOM, a source is connected to its receivers by an *overlay Steiner tree*. At the overlay level, packets are multicast to all receivers along the Steiner tree. In the underlying wireless network, packet transmission on each overlay link is realized by a multi-hop NC-based OR transmission. The unique features of NCOM is summarized as the following:

- Different from MORE and Pacifier, the multicast topology of NCOM explicitly takes into account the cost of OR transmissions between nodes. It can more effectively merge the OR relay paths to different receivers and maximally reduce the number of transmissions required for multicast.
- MORE and Pacifier calculate the expected number of transmissions for each forwarders based on periodic link state measurement. This leads to significant performance loss in dynamic wireless environment. NCOM employs a cumulative coded acknowledgement scheme customized for multicast to coordinate the transmissions of forwarders. It automatically adapts the number of transmissions of each forwarder to the actual packet losses. It also enables opportunistic packet reception cross adjacent overlay links.
- In NCOM, the medium access of conflicting transmissions is weighted by the “multicast information gain” of each transmission. Transmissions beneficial for more receivers are given higher priority. It allows NCOM efficiently schedule wireless links to achieve high multicast throughput. It also naturally generates back-pressure to control the transmission redundancy of the forwarders and the source.

We implement NCOM in OPNET by customizing the IEEE 802.11b modules. Through OPNET simulations, we demonstrate that NCOM can achieve a higher throughput and lower source transmission redundancy than the existing NC and OR based wireless multicast designs.

The rest of the paper is organized as follows. We briefly review the related work in Section II. The NCOM scheme is presented in Section III. The NCOM protocol design and implementation in OPNET is presented in Section IV. The performance of NCOM is evaluated through OPNET simulations in Section V. The paper is concluded in Sections VI,VII.

II. Related work

The original Opportunistic Routing algorithm, called ExOR was first proposed by S. Biswas and R. Morris in [6]. Instead of pre-selecting a multi-hop path, ExOR exploits the broadcast nature of wireless transmission and employs a dynamic sequence of forwarders to deliver a packet to its destination. In [16], the authors introduced a robust distributed opportunistic routing scheme base on ETX metric that can find the optimal OR path from a source to a receiver. Authors of [17] conducted a systematic performance evaluation of OR by taking into account node densities, channel qualities and traffic rates to identify the cases when OR makes sense. The work in [18] proposed a method to calculate the maximum throughput between two end nodes with OR in ad-hoc networks. The recent work from [19] studies the capacity of hybrid wireless networks under OR, which exploits high speed data transmissions in infrastructure network through base stations to improve the routing performance. In [20],the authors introduced CodeOR protocol, which deploys network coding by transmitting a window of multiple segments concurrently in opportunistic routing to improve throughput. The work in [21] proposed the SlideOR protocol in which the source node uses a moving sliding window to determine the set of packets to transmit without segmentation. This new network coding technique together with OR is proved to have higher performance, and is simpler to implement than the previous OR works. The most recent work on unicast OR is [22], which deploys piggyback ACKs inside each data packet to coordinate transmissions on forwarding nodes. The CCACK protocol incurs less transmission redundancy and can achieve higher throughput than MORE [3].

III. NCOM Design

In this section, we present the NCOM design. We start with our network assumptions and an architecture overview of NCOM. We then present three major design components of NCOM: OR-based overlay multicast tree, NC-based OR transmission along overlay links, and cross-layer link scheduling.

A. Network Model and NCOM Overview

We consider a network of N static wireless nodes, including one source node S, a set of $K < N$ receivers $\mathcal{R} = \{R_1, R_2, ..., R_K\}$, and $N - K - 1$ relay nodes. All nodes are equipped with radio interfaces and can communicate with neighbor nodes within their effective radio transmission ranges. Wireless links between neighbor nodes are not

reliable. The success probability of packet transmission on a link is given by the Packet Reception Ratio (PRR). The PRR p_{ij} of link $\langle i, j \rangle$ theoretically depends on the distance between nodes i and j, node density and traffic around i and j, and the MAC scheduling scheme. As commonly assumed [23], packet losses on different links are independent. At a high level, NCOM is an overlay multicast scheme developed for wireless networks. Overlay

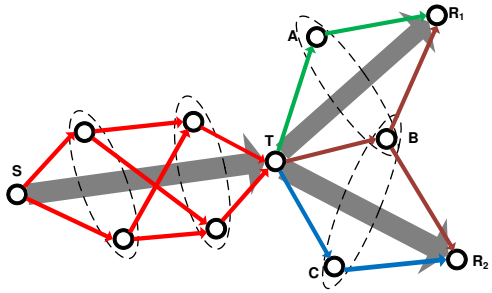


Fig. 1. Conceptual View of NCOM.

networks have been widely employed to deliver multicast services on the Internet [24], [25]. Overlay multicast does not require native network multicast support, and can be easily deployed based on unicast primitives. In NCOM, given PRR on wireless links, an overlay Steiner tree is first established to connect the source with all receivers. The overlay tree serves as a virtual multicast backbone. At the overlay level, packets are multicast to all receivers along the overlay tree. Neighbors in the overlay tree are not necessarily neighbors in the underlying wireless network. Packet transmissions on each overlay link are realized by multi-hop NC-based OR transmissions in the underlying wireless network.

Figure 1 illustrates an example of NCOM overlay tree with one source S and two receivers R_1, R_2, and one overlay Steiner node T. The thick arrows are the overlay links. The rest of nodes in the figure are OR forwarders on overlay links. The thin arrows are the potential NC-based OR transmissions in the underlying wireless network along overlay links.

B. OR-based Overlay Multicast Tree

The overlay tree is constructed to minimize the number of OR transmissions to disseminate data to all receivers.

1) Unicast OR Distance between Nodes: Given a wireless network, the length of a unicast OR path from node i to node j is the expected number of packet transmissions to send a packet from i to j along the OR path. It is a function of the PRR on all links. The OR distance from i to j is defined as the length of the shortest OR path from i to j. In the recent work from Laufer et al. [26], they proposed the Shortest Anypath First (SAF) and Shortest Multi-rate Anypath First (SMAF) algorithms to calculate the optimal OR paths from every node in a network to one receiver with single and multiple transmission rates. These schemes were shown to have the same complexity of Dijkstra's shortest path algorithm.

Algorithm 1 Minimum Steiner Tree Algorithm

```
X ⇐ Y
C = MST(X)
repeat
    X_temp ⇐ X
    i* = ∞
    for i ∈ V \ X do
        C_temp = MST(X ∪ i)
        if C_temp < C then
            C = C_temp
            i* = i
        end if
    end for
    if i* ≠ ∞ then
        X ⇐ X ∪ i*
    end if
until (X ≠ X_temp)
```

2) Minimum Steiner Tree algorithm: We assume the wireless network is connected and any pair of nodes can reach each other using unicast OR. Given p_{ij} on all links, every node i calculates the shortest OR distance L_{ij} to any other node j based on the SAF algorithm developed in [26]. We construct a fully connected overlay network $G^o = (V, E^o)$, with V consisting of all nodes in the network and $E^o = V \times V$. The cost of the virtual link between i and j is the OR distance L_{ij} in the underlying wireless network. In the overlay graph, we construct the minimum Steiner tree connecting the source node S with the set of receivers $\mathcal{R}$. The most popular algorithm to construct the minimum Steiner tree was proposed by Dreyfus and Wagner [27] based on dynamic programming. The complexity of the algorithm is $O^*(3^k)$, where k is the number of terminal nodes to be connected. However, the dynamic nature of wireless ad-hoc networks requires a simpler minimum Steiner tree algorithm. In the following, we propose a simple heuristic algorithm to calculate a Steiner tree for our purpose.

Let $Y = S \cup \mathcal{R}$ be the set of terminal nodes. The other nodes on the tree called Steiner nodes. The idea for the heuristic algorithm is to first construct the Minimum Spanning Tree among all terminal nodes, then grow the spanning tree into a Steiner tree by incorporating Steiner nodes step by step. At each step, the algorithm will find a node i^*, that when added to the tree could maximize the reduction of the total cost of the current tree. In Algorithm 1, Set X stands for the set of nodes that have been admitted into the Steiner tree. MST(X) is the cost of the Minimum Spanning Tree of the set X. The fastest minimum spanning tree algorithm to date can calculate

MST(X) in close to linear time [28]. Variable C stores the cost of the minimum Steiner tree. The algorithm completes when the cost could not be further reduced. This heuristic algorithm takes $O(V^2E)$ time to complete.

C. NC-based OR Transmissions along Overlay Links

With the established overlay Steiner tree, packets are transmitted along the overlay links to reach all receivers. With OR, nodes located around overlay links can serve as candidate relay nodes to forward packets from one overlay node to its downstream node in the overlay tree. The transmission along an overlay link (i, j) can simply be implemented as an unicast OR flow, similar to [3] and [22], in the underlying wireless network, with i be the unicast source and j be the unicast destination. For example in Figure 1, S establishes one unicast OR flow and sends packets to the Steiner node T. T could establish two separate OR unicast flows, one for overlay link (T, R_1), the other for (T, R_2). However, such a design does not fully exploit the broadcast nature of wireless transmission and the opportunistic packet reception cross adjacent overlay links. In the example of Figure 1, due to its unique location, node B is T's candidate forwarder to both receivers R_1 and R_2. After one transmission from T, if B receives an innovative packet, it can potentially broadcast the packet simultaneously to R_1 and R_2. In the best case, only two transmissions are needed for T to send a packet to R_1 and R_2. For comparison, if T employs two separate unicast OR flows, at least four transmissions, two on each overlay link, are needed. In NCOM, we design NC-based OR Transmissions along overlay links to maximally take the OR gain.

1) Candidate Receiver Set: In unicast OR, each node finds nodes with shorter distances to the unicast destination as its candidate forwarders. In the overlay tree of NCOM, each overlay node (except for the multicast source S) is a potential receiver of OR transmissions. Consequently, a node can be chosen as a forwarder for multiple receivers on adjacent overlay links. Algorithm 2 - "Candidate Receiver Set Algorithm" is used to calculate the set of candidate receivers $\mathcal{D}_i$ towards which node i should forward innovative packets that it received from its upstream nodes along the overlay links. In Algorithm 2, S and D is an overlay sender-receiver pair on the overlay link (S, D). The variables M_{ij} are used to mark if node i already get node j in its Candidate Receiver Set (CRS). CFS_S^D is the Candidate Forwarding Set of node S towards receiver D. It includes every node that is located in the effective radio range of node S, and has a smaller OR distance to D than S. V is the set of nodes in the network. $E^o_{Steiner}$ is the set of overlay links of the overlay Steiner tree. In Function: *CRSA(S,D)*, every node i that potentially join the OR routing path from S to D will add D to its CRS $\mathcal{D}_i$. Algorithm 2 scans all overlay links in the overlay Steiner tree and gradually adds candidate receiver nodes to the CRS $\mathcal{D}_i$ of every node i. When Algorithm 2 finishes, every node will find its CRS. We let $L_i = |\mathcal{D}_i|$ be the size of node i's CRS.

2) Multicast Cumulative Coded Acknowledgments - MCCACK: One challenge of OR is how to coordinate transmissions of forwarders to guarantee reliable data delivery without incur high data redundancy. Specifically, a forwarder needs to determine which received packets it should forward, and at what rate. By employing NC [22], [3], [4], the coordination between forwarders can be significantly simplified. With NC, packets are randomly mixed

Algorithm 2 Candidate Receiver Set Algorithm

Function: *CRSA(S,D)*
$\mathcal{D}_S \Leftarrow D$
$M_{SD} \leftarrow TRUE$
for $i \in CFS_S^D$ **do**
  if $M_{iD} = FALSE$ **then**
    CRSA(i,D)
  end if
end for
end Function

Main Program:
$M_{ij} \leftarrow FALSE \quad \forall i, j \in V$
$\mathcal{D}_i \Leftarrow \emptyset \quad \forall i \in V$
for $(i, j) \in E^o_{Steiner}$ **do**
  CRSA(i,j)
end for
end Main Program

at the source and forwarders. Both MORE and Pacifier compute offline the number of expected transmissions for each forwarder using heuristic based on periodic measurement of link loss rate. Then innovative coded packets are transmitted by forwarders at the pre-computed rate. Unfortunately, such an "open-loop" design leads to significant performance loss if the link measurement is not accurate or the link losses are dynamic.

In [22], Koutsonikolas et al. proposed to use Cumulative Coded Acknowledgments (CCACK) between forwarders to coordinate forwarder transmissions. In CCACK, a node piggybacks a specially designed ACK vector in its data packets to inform its upstream nodes the received coded packets. Based on such packet reception feedback, upstream nodes make a decision on whether temporarily or permanently stop sending coded packets. The "close-loop" design of CCACK makes it robust against dynamic packet losses. It was shown in [22] that CCACK can significantly reduce source redundancy and achieve high throughput gain. CCACK is designed for unicast. In NCOM, we customize CCACK to fully take advantage of OR in the multicast case.

We start with a brief introduction of CCACK. In CCACK, every node maintains three different queue structures in order to estimate how many innovative coded packets that a node could provide for its downstream nodes. When a node has zero innovative coded packet for its downstream nodes, it will temporarily stop sending until it get new innovative packets from its upstream nodes. On a node i, queue B_v stores only innovative coded packets that i received from its upstream node. Queue B_u stores the coefficient vectors of the coded packets (no need to be

innovative) successfully received by i from its upstream nodes. Queue B_w stores the coefficient vectors of the packets that i sends out. When it is the turn for i to send out a data packet, it generates a coded packet through a random linear combination of packets in its queue B_v and sends it out. Since the packet will be overheard by i's upstream nodes, i also piggyback a ACK vector to inform its upstream nodes about the coded packets in its queue B_u. To improve the efficiency of the ACK and reduce the header size, i does not directly piggyback the coefficient vector of each packet in its queue B_u. Instead, only one special ACK vector $\vec{z}$ is generated and transmitted. $\vec{z}$ is orthogonal to a matrix Δ created from a group of vectors u in i's queue B_u and M hash matrices $\{H_k^{(i)}, 1 \leq k \leq M\}$ unique to node i. When an upstream node of i, say j, receives the ACK vector $\vec{z}$ piggybacked in a data packet sent out from i, it can detect which vectors in its B_u and B_w queues have been received by i. Specifically, on node j, if Equation 1 is satisfied, it is inferred that the coded packet with coding vector u has been received by i.

$$u \times H_k^{(i)} \times \vec{z}^T = 0, u \in B_u \cup B_w, \forall k = 1, ..., M, \tag{1}$$

The coding vector u then is marked as H by node j. If the rank of marked vectors in $B_u \cup B_w$ is equal to the rank of B_v, j will stop transmitting coded packet. That is because the upstream node detects that it temporarily does not have any innovative packet beneficial for its downstream nodes. More details about CCACK can be found in [22]

CCACK only works with single unicast destination. To customize CCACK for overlay multicast, one has to deal with the case that a forwarder might be responsible for forwarding coded packets to multiple receivers. In MCCACK, every node i maintains three different queue structures similar to CCACK:

B_v stores innovative coded packets for every wireless node. Note that "innovative" is the only criterion for a packet to be stored in B_v. The packet could come from any node around i, no mater it is a upstream or downstream node of i. Even if i overheard an innovative packet for a receiver k from one of its downstream node towards k, i still puts that packet into its queue B_v. Because this packet might be innovative for i's candidate receivers other than k. This is different from CCACK where B_v only store the innovative coded packets coming from upstream nodes toward the receiver.

B_u stores the coefficient vectors of the coded packets received by i from its upstream nodes. For each receiver k in the CRS $\mathcal{D}_i$, i has one separate queue B_u^k. When i gets a packet from an upstream node towards receiver k, it will store the coefficient vector of that packet in B_u^k. For a received packet, i might update multiple queues B_u^k if the upstream node has i as forwarder for multiple receivers. (In Figure 1, Node B updates $B_u^{R_1}$ and $B_u^{R_2}$ upon receiving an innovative packet from T).

B_w stores the coefficient vectors of packets sent out by node i. For each receiver k in the CRS $\mathcal{D}_i$, i has one separate queue B_w^k. However, everytime i send out a packet, it always add the coefficient vector of that packet in every queue B_w^k, $\forall k$ in the CRS $\mathcal{D}_i$. We need different queues B_w^k because they help to clarify for each receiver k

$\in$ CRS $\mathcal{D}_i$, how many innovative packets i could still supply for CFS_i^k.

Node i generates one separate ACK vector for each candidate receiver $k \in \mathcal{D}_i$ to inform its upstream nodes about the packets inside its B_u^k queues. Node i applies CCACK's ACK generation algorithm to the B_u^k queue to generate the ACK vector for receiver k. Each time node i transmits a data packet, it piggybacks $L_i = |\mathcal{D}_i|$ ACK vectors. (L_i in practice is small, normally 3 or less). On an upstream node of i, say j, upon receiving i's ACK vector towards receiver k, it applies CCACK's vector detection algorithm to identify and mark the received coding vectors in $B_u^k \cup B_w^k$. j does the same operation for ACK feedback received from all of its candidate forwarders towards k. On node j, let B_H^k be the set of vectors in $B_u^k \cup B_w^k$ marked as H. Call:

$$q_j^k = dim(B_v) - dim(B_H^k) \tag{2}$$

In Equation 2, q_j^k is the difference between the number of innovative packets at node j and the aggregate number of innovative packets available on nodes in its Candidate Forwarding Set toward receiver k - CFS_j^k. IF $q_j^k \leq 0$, node j does not have any innovative packet for his candidate forwarders towards k, it should suspend the transmission towards k. Therefore, whenever it is the turn for j to send out one data packet, it only transmits towards receivers $k \in \mathcal{D}_j$ such that $q_j^k > 0$ and $dim(B_H^k) < N$. If no such receivers exists, j just keep silent and wait for new innovative packets from upstream nodes.

For every receiver, it broadcasts a ACK packet to its upstream nodes whenever received a code data packet sent to it, no mater if that packet is innovative or not. The ACK packets sent out without data payload since it sent from receiver nodes. This is necessary to inform its upstream nodes whether they should temporarily stop transmitting.

D. Cross-layer Link Scheduling

MCCACK provides feedback for nodes to determine when they should stop transmitting packets for a given batch, but it does not say anything about how fast nodes should transmit before they stop. In MORE [3], Pacifier [4], a downstream node is triggered to transmit coded packets by receptions from upstream nodes. The sending rate of a node is controlled by the pre-computed transmission credit. The CCACK protocol [22] also uses a simple credit scheme, which is oblivious to loss rates but aware of the existence of other flows in the neighborhood. All the above schemes assume a given link level scheduling, such as the standard IEEE 802.11 MAC, do not have direct control on the wireless channel access.

It has been theoretically shown that the network capacity can be achieved through cross-layer operations involving source rate control, network routing, and link scheduling [7], [8], [9], [11], [10]. In a pioneer work [29] for wireless unicast, the Maximum Weight Matching (MWM) type of wireless link scheduling is proved to be throughput-optimal. In MWM, each wireless link is weighted by the queue backlog difference between the link transmitter and receiver. The optimal link schedule can be obtained by solving a maximum weight matching problem. This work has recently been extended to study the optimal cross-layer scheduling for multicast flows with network coding and broadcast

wireless links [12], [13], [14]. It was theoretically shown that a broadcast link can be weighted by the summation of the "information gain" towards all multicast receivers, and the throughput-optimal cross-layer scheduling can be obtained by solving a Maximum Information Weight Matching (MIWM) problem [14]. However, it is NP-Hard to exactly solve the MIWM problem.

Motivated by the MIWM multicast link scheduling, we design a simple cross-layer link scheduling scheme for NCOM to control the wireless channel access for high wireless multicast efficiency. We first quantify the potential number of innovative packets q_i^k, that a node i could provide for its CFS_i^k toward receiver k:

$$q_i^k = dim(B_v) - dim(B_H^k) \tag{3}$$

Then the "information gain" of node i toward receiver k is modeled as:

$$\Delta Q_i^k = \max_{j \in CFS_i^k} [q_i^k - q_j^k]^+ \tag{4}$$

The "information gain" of node i towards all potential receivers is:

$$\Delta Q_i = \sum_{k \in \mathcal{D}_i} \Delta Q_i^k \tag{5}$$

In Equation 3, $dim(B_v)$ is the number of innovative packets in queue B_v of node i. $dim(B_H^k)$ is the number of innovative packets marked as H in the queue B_H^k towards receiver $k \in \mathcal{D}_i$. So q_i^k is the potential number of innovative packets that node i could provide for its CFS_i^k toward receiver k.

In Equation 4, ΔQ_i^k is the biggest information queue length difference between node i and every node $j \in CFS_i^k$. It mean the "information gain" of node i toward receiver k. Equation 5 is the summation of all "information gain" of node i toward every receiver in its Candidate Receiver Set $\mathcal{D}_i$. ΔQ_i could be deployed as the weight of the broadcast link of node i in MIWM problem.

Following the spirit of MIWM type of cross-layer scheduling, nodes i with higher value of ΔQ_i will have higher probability to access the channel. NCOM employs a CSMA/CA type of distributed MAC layer design. On each node, we use its information gain ΔQ to regulate the packet delay time G in CSMA/CA. We set $G = G_1 + G_2$ such that, $G_1 = F(\Delta Q), G_2 = Random(0, \alpha_{max})$. $F(\cdot)$ is a decreasing function such that nodes with larger information gain ΔQ have higher priority in accessing the channel. G_2 is used to keep the randomness of G to desynchronize the collision avoidance on competing nodes. We will elaborate the implementation of the proposed link scheduling in the following section.

IV. NCOM OPNET Implementation

A. Simulation Setup

To test the performance of NCOM, we implement the NCOM protocol in OPNET Modeler Version 14.0 by customizing the IEEE 802.11b wireless local area network simulation modules. We inherited the implementation

of the physical layer and the data link layer of the IEEE 802.11b from the OPNET standard library. NCOM is implemented by customizing the application layer, network layer and MAC layer of IEEE 802.11b.

The same as MORE [3] and Pacifier [4], node could deploy the bootstrap progress to measure the average packet loss rates on wireless links. That values will be flooded all over the network. Every node will deploy the simple Algorithm 1 to calculate the overlay Steiner tree and Algorithm 2 to find out if it could involve in the OR transmission progress. Nodes will self-calculate their Candidate Receiver Set and Candidate Forwarding Set toward each of their candidate receiver node. After the initial overlay Steiner tree construction, an overlay node knows its parent and children in the overlay tree. Whenever a node receives a packet from its parent, it will forward the packet to its overlay children using multicast OR. When a node sends a coded packet to its downstream nodes in the overlay Steiner tree, we temporarily refer the node as the sender and the downstream nodes as the receivers for this multicast OR. Similar to MORE and Pacifier, we implement multicast OR with network coding. Specifically, the original data file is divided into batches of eight packets. If the sender is the source node, coded packets are generated by random linear combination of the original data packets. If the sender is a forwarder node in the overlay tree, coded packets are generated by random linear combination of innovative coded packets it received and stored in queue B_v. All coded packets are generated from the packets in the same batch. Receivers can decode a batch of original packets upon receiving eight independent coded packets from the same batch.

The sender knows the receivers' MAC addresses and the OR distances from its neighbors to the receivers. It constructs the candidate forwarding sets (CFS)s consisting of neighbors that have shorter OR distances toward the receivers than itself. The MAC addresses of forwarders in CFSs toward each receiver are embedded in the header of the packet sent out.

As described in Section III-D, to help MAC layer determines the contention windows size for a packet transmission, the application layer calculates ΔQ and embeds this information in the packet header. The ACK vectors towards each receiver is also calculated and embedded in the header at application layer. Whenever sending out a data packet, a node always piggyback every ACK vector towards each receiver in its CRS.

In our experiment, multicast OR will be deployed on top of IEEE802.11b Wireless LAN Standard. At MAC layer, packet send out in broadcast mode. The CSMA/CA mechanism with disabled option of CTS/RTS will be deployed. To avoid the collision, during the duration that medium is idle, node waits for a random number of timeslots in the range from 0 to the Contention Window (CW) before attempting to assess a channel. CW is the contention window length, defining the number of timeslots that a channel needs to be idle before a transmission can take place. This value is initialized before each transmission attempt. The CW counter will be on hold when the channel is assessed to be busy and resume to count when channel return to idle . Nodes with higher value of ΔQ will have smaller value of contention window size CW.

For NCOM implementation without crosslayer scheduling, the standard initial value of CW = 31 timeslots. The

duration of a slottime is 20 μs. With crosslayer scheduling, the initial value of CW will be dynamically changed based on the value of ΔQ as described in Section III-D. With each node i

$$CW = \left\lceil \frac{31}{\Delta Q_i} \right\rceil \tag{6}$$

On the receiving side, whenever a node receives a packet from MAC layer, it analyzes the header to see whether the packet contains piggybacked MCCACK vectors or coded data. At the application layer, it then process the coded data or the MCCACK vectors to update queues B_v, B_u^k and set B_H^k for every receiver k in its CRS. It also updates the value of ΔQ. If a node is a multicast destination, whenever it receives enough innovative coded packets from the same batch, it can recover the original packets in this batch. It then informs the source about the transmission process for the current batch is completed. When the source detects every destination successfully received the data of the current batch, it will move to send out the data of the next batch. If a node is a relay node and got a packet with a new batch ID, it will clear all the queues of the current batch and start to process the new batch. A node is triggered to send out data if it has innovative packets to benefit its downstream nodes toward any receiver k in its CRS. A node stops data transmission if it has ΔQ = 0. This process continues until all packets of the data file are delivered to all destination nodes.

V. Performance Evaluation

To evaluate the performance of NCOM, we implemented the following multicast protocols using the IEEE 802.11b wireless LAN network simulation modules.

- Pacifier protocol
- NCOM with MCCACK
- NCOM with crosslayer scheduling and MCCACK

Since Pacifier achieves higher throughput and lower transmission redundancy than MORE [4], we didn't compare NCOM with MORE.

In Pacifier, source constructs a multicast tree by merging the shortest paths (based on the ETX metric) to all receivers. Coded packets are forwarded along the multicast tree to receivers. One component of the Pacifier is the round robin batch selection. Pacifier moves to the next batch when one receiver acknowledges the completion of receiving the current batch. Source node keeps repeating the uncompleted batches in the round robin fashion until all receivers successfully received all batches. That will reduce the influence of the well-known "crying-baby" problem when one receiver with poor connectivity slowing down the performance of the whole multicast group. However that scheme will made the batch latency is very high for many receivers and not suitable for the delay sensitive applications.

The "crying baby" issue has common influences to every multicast routing scheme and the proposed method of Pacifier could apply to all routing scheme like MORE, Pacifier or NCOM with the trade-off advantages on

throughput and delay. To get the direct comparison on the performance of the multicast Oppotunistic Routing efficiency, we do not implement the "crying baby" solution for Pacifier and NCOM in our experiments.

We then evaluate the performance of the NCOM with MCCACK. The OR transmissions and piggyback MCCACK mechanism are fully multicast. Lastly, we turn on the cross-layer link scheduling for NCOM for additional performance improvement.

In the OPNET IEEE 802.11b, the sending rate at the physical layer is set to be 11Mb/s. We configure the transmission power with the effective radio coverage of 250m.

A. Simulation Results and Analysis

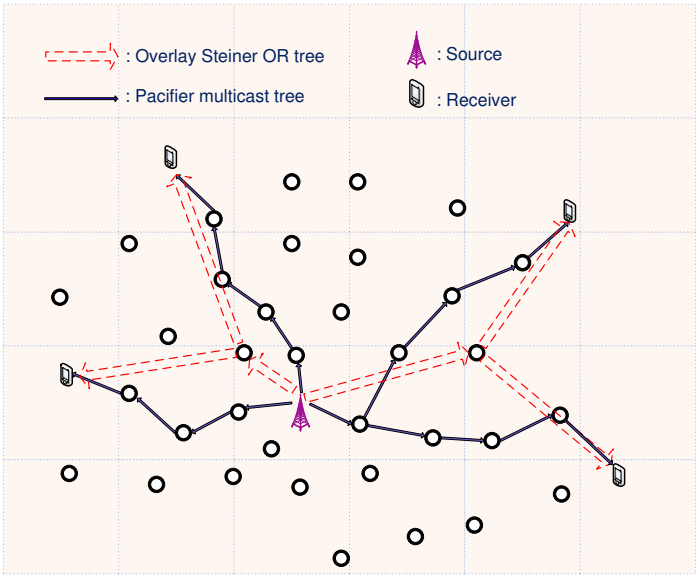
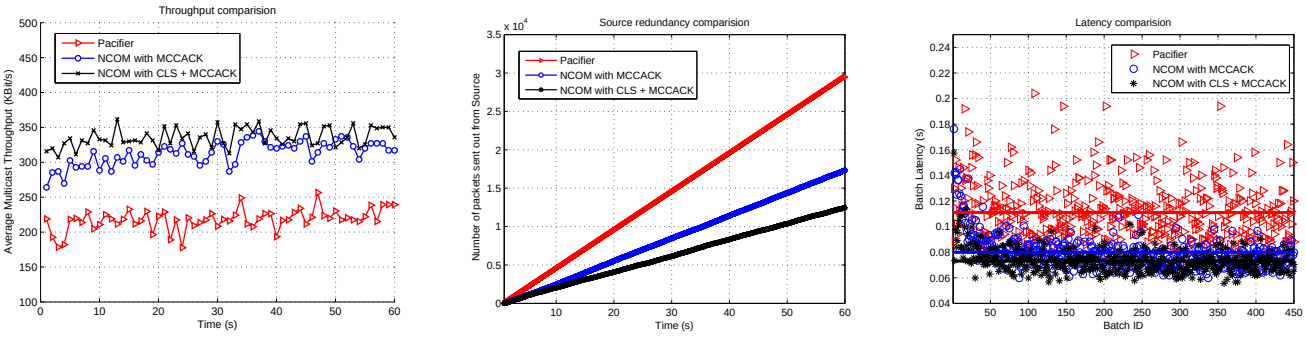


Fig. 2. Simulated Scenarios One: topology of 40-nodes network, Pacifier Multicast Tree, and Overlay Steiner Tree.

We first study a network with 40 static nodes randomly located in an area of $1000m \times 1500m$ as in Figure 2. The grid size is $250m$. We randomly choose one source and four receivers. The average packet loss rate on each wireless link is around 10%, which is determined by the distance between nodes, physical layer setting and data link layer scheduling scheme. The source sends multicast traffic to receivers with three different routing schemes as referred in Section IV-A. For Pacifier, the multicast tree based on the ETX metric is plotted as the thin arrows. For NCOM routing schemes, the overlay Steiner tree based on the OR distance is plotted as the thick arrows. Routes from the source to receivers ranged from 4 to 5 hops.

Figure 3 presents the comparison of throughput gain, source redundancy and batch latency on three routing schemes. Figure 3(a) plots the average throughput delivered at each receiver on every second as the simulations progress. From the result, the average multicast throughput with Pacifier is the lowest. NCOM with MCCACK is 43% higher than Pacifier. NCOM with CLS + MCCACK get approximately 54% throughput gain over Pacifier. Crosslayer scheduling further improves the throughput of NCOM with MCCACK by almost 10%. The throughput



(a) Throughput comparison (b) Traffic from source comparison (c) Data Latency comparison

Fig. 3. Performance Comparisons between NCOM and Pacifier in one 40-nodes Topology

improvement of NCOM routing schemes over Pacifier is firstly credited to the minimum overlay Steiner tree based on OR distance, compared with the Pacifier multicast tree. The design of Pacifier's multicast tree is compromised by using the unicast ETX distance between nodes to guide OR transmissions between nodes. Nodes around the tree do not participate the data transmission. For NCOM overlay Steiner tree, the OR distance used to calculate the tree is more accurately reflex the transmission costs between nodes.

Secondly, throughput gain can come from the MCCACK, which is customized from CCACK to work with multicast OR. The significant thoughput gain from NCOM with MCCACK over NCOM with CCACK demonstrate the advantage of multicast OR over unicast OR along the overlay tree. MCCACK reduces the number of redundant transmissions cross adjacent overlay links, and hence improve the multicast throughput. Thirdly, with crosslayer scheduling, nodes with more innovative packets to benefit for its downstream nodes can send packets faster. This helps NCOM more efficiently schedule link transmissions and achieve an additional 11% throughput improvement in this scenario.

Figure 3(b) plots the number of coded packets that the source sends out with each routing scheme. These values together with the throughput results in Figure 3(a) can be used to calculate the average source redundancies as in Table I. NCOM routing schemes got much lower source redundancies than Pacifier because of the MCCACK deployment. MCCACK mechanism helps to significantly reduce the number of redundant packets sent out from source. Source redundancy of NCOM with CLS + MCCACK is only 2.16 while Pacifier get more than 3 times higher than that with 6.79. Due to the backpressure effect, cross layer scheduling helps the source quickly adjusts its sending rate to match the network congestion, so NCOM with CLS + MCCACK gets even lower source redundancy compare to NCOM with MCCACK.

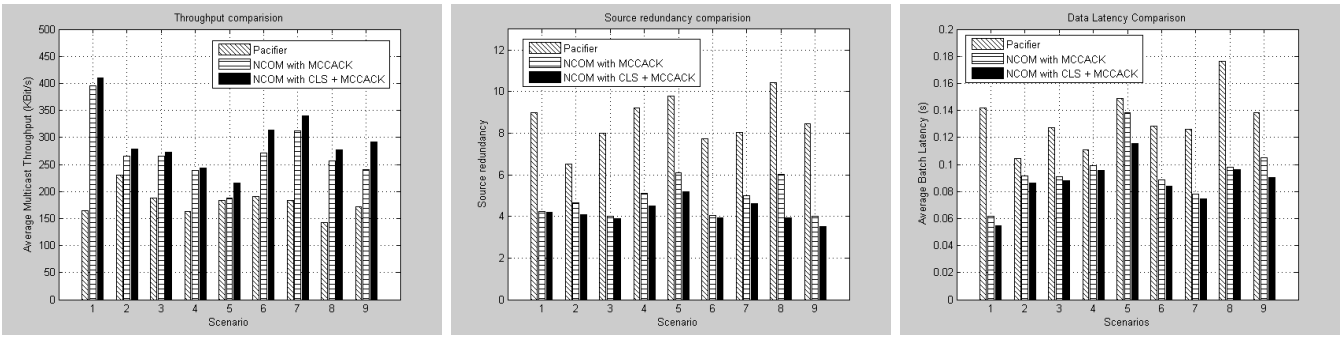
We also measured the latency for each batch of packets. Batch latency is defined as the time lag from the source sends out the first packet of the batch until it gets the information that all receivers get enough number of coded packets to decode that original batch of data. Figure 3(c) plots the latency of each batch of data with all three routing schemes. The horizontal lines are the mean values of batch latencies. The values are also given in the Table

	Source redundancy	Average Batch latency(s)
Pacifier	6.79	0.111
NCOM with MCCACK	3.24	0.0799
NCOM with CLS+MCCACK	2.16	0.0731

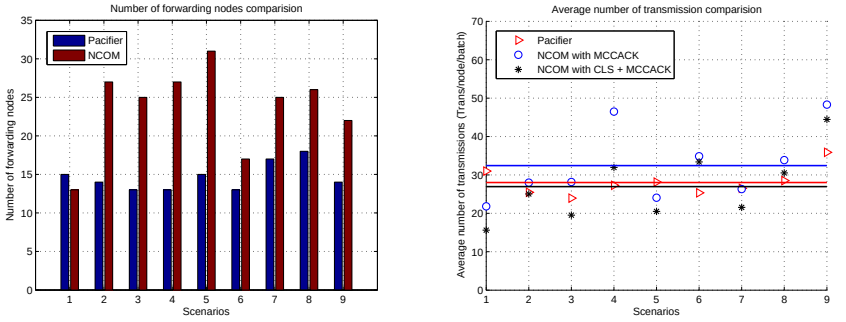
TABLE I

REDUNDANCY AND LATENCY COMPARISONS

I. Consistent with the throughput comparison, NCOM with CLS + MCCACK gets the lowest average batch latency with 0.0731s, while Pacifier needs on average 0.111s to complete the transmission of one batch of data from the source to receivers. NCOM with MCCACK has a little higher Batch latency (0.0799s) than NCOM with CLS + MCCACK.



(a) Throughput comparisons (b) Source redundancy comparisons (c) Data Latency comparison



(d) Number of FNs comparisons (e) Number of transmission comparisons

Fig. 4. Performance Comparisons under Nine Different Network Scenarios

We then compare the performance of NCOM routing schemes with Pacifier in more scenarios. With the set of 40 nodes randomly located in an area of $1000m \times 1500m$, we randomly choose one source and five receivers for nine times. Each time, we run the scenario with three routing schemes: Pacifier, NCOM with MCCACK and NCOM with CLS + MCCACK.

Figure 4 presents the comparisons of the three schemes under nine scenarios. Table II lists the average values of throughput gain, source redundancy and batch latency in three routing schemes. In Figure 4(a), NCOM routing

	Average throughput (Kbps)	Average source redundancy	Average Batch latency(s)
Pacifier	179.720	8.574	0.141
NCOM with MCCACK	270.420	4.797	0.094
NCOM with CLS + MCCACK	293.544	4.208	0.087

TABLE II

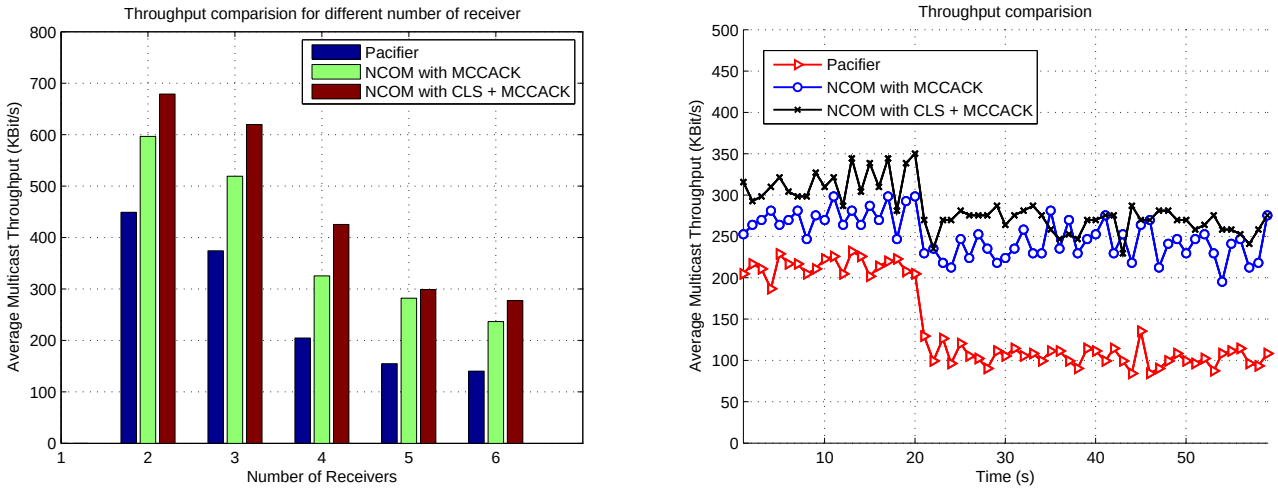
COMPARISONS ON 9 SCENARIOS

schemes always get higher throughput than Pacifier. The gain ranges from 17.3% (Scenario 5) up to 249% (Scenario 1), with an average throughput gain of NCOM with CLS + MCCACK is around 63% higher than Pacifier. The throughput of NCOM with CLS + MCCACK is consistently higher than NCOM + MCCACK with the average gain about 8.5%. Thanks for the contribution from cross-layer link scheduling. Looking into the details of the simulation data, NCOM gets bigger gain over Pacifier in the scenarios with higher network density and the difference between the cost (based on ETX metric) of the Pacifier's tree and NCOM's overlay tree get bigger. In those scenarios, the overlay minimum Steiner tree will get higher advantage over the multicast tree of Pacifier. The overlay Steiner tree could find a better path to the receivers since it uses more relay nodes to forward the packets. Vice versa, in the scenario when one receiver farther from the source with low node density along the path when all other receivers are very near the source, the performance difference between NCOM and Pacifier get smaller.

Figure 4(b) demonstrates the source redundancy comparisons of the three routing schemes over the nine scenarios. NCOM with CLS + MCCACK always gets the lowest source redundancy for all the cases with the average value of 4.208. That is just less than 50% of source redundancy of Pacifier (8.574). Without cross-layer scheduling, the source redundancy of NCOM with MCCACK gets to approximately 56% of Pacifier. The gain mostly comes from the piggyback MCCACK mechanism. MCCACK + CLS could reduce source redundancy more because the backpressure process working more efficiently from the receivers back to the source node. Figure 4(c) presents the batch latency comparisons over nine scenarios. NCOM with CLS + MCCACK gets the lowest latency, with only 0.087s compares to 0.141s of Pacifier. Cross-layer scheduling on average could reduce the average batch latency by 0.007s. That is equivalent to 8% of batch latency of NCOM with MCCACK + CLS.

We also looks at the number of nodes involved in the forwarding progress in Figure 4(d). NCOM with and without CLS have the same number of forwarding nodes involed in the OR transmission progress. The numbers of forwarding nodes in NCOMs are higher than Pacifier in most of the cases. On average, the average number of forwarding nodes in the 9 scenarios of NCOMs is equivalent to 1.614 times that number of Pacifier (23.67 to 14.67). The average number of transmissions/ forwarding node / batch of NCOM with CLS + MCCACK is a little smaller than Pacifier (26.952 compare to 28.017) (Figure 4(e)). The measurements above showed that, per each scenario, the total number of transmissions of NCOM with CLS + MCCACK is around 60% higher than that

number of Pacifier. That cost will payback with 63% throughput gain of NCOM with CLS + MCCACK compare to Pacifier. That means the heuristic based tree build up of Pacifier is too conservative with too few forwarders in a sparse network. NCOM with CLS + MCCACK gets higher throughput with more nodes involved in transmission progress. Figure 4(e) also showed that CLS could help NCOM to reduce the average number of transmissions/node by around 20% while improve throughput by 8.5%.



(a) Different number of receiver comparison

(b) Resilience to dynamic packet loss rate

Fig. 5. Performance Comparisons between NCOM and Pacifier

In Figure 5(a), we try to measure the performance of NCOM and Pacifier with the different number of receivers. We set the cases with 40 nodes and change the number of receivers from two to six. We then measure the average multicast throughput for all three routing mechanisms. Figure 5(a) presents the throughput comparison. The throughput of NCOM with CLS + MCCACK consistently higher than Pacifier on average 69%. With 2 receivers , the throughput gain is around 50% and increased to more than 100% in the case of 6 receivers. With the more receivers, the average throughput difference between NCOM and Pacifier get higher. That is because the overlay tree of NCOM gets more advantage compare to the Pacifier's tree.

We finally keep the case with 4 receivers and simulate the case when packet loss rates of the wireless links dynamically changed in bound $\pm$50% every 5s after 20s of the simulation progress. Figure 5(b) shows that NCOM get much higher resilience to dynamic packet loss rate with 14% throughput reduction compare to 50% of Pacifier. Thanks to the advantage of the overlay Steiner tree over multicast tree of Pacifier. Pacifier have a significant throughput reduction since its deploy a static physical multicast tree, which is more sensitive than NCOM's tree on dynamic packet loss rates. Overlay Steiner tree is more robust because it deploys more relay nodes when forwarding packets along the overlay link. The performance may less depend on the packet loss rate on a specific wireless link. More than that, MCCACK is also robust on wireless link loss rate since it deploy "online" link loss rates to control the start/stop transmission process at each wireless node. All on all, those mechanisms help NCOM more

robust on dynamic packet loss rate on wireless network. This is a very critical advantage of NCOM over Pacifier.

VI. Acknowledgments

We would like to deeply thanks Professor Y. Charlie Hu and Dimitrios Koutsonikolas at School of Electrical and Computer Engineering, Purdue University for exchanging and discussing with us on Pacifier and CCACK works. We got a great help from the Pacifier simulation sharing from them. That we could learned and implemented on OPNET.

We also would like to thanks Yuting Zheng, Chiang Liu, Sha Hua and Xiwang Yang from Department of Electrical and Computer Engineering, Polytechnic Institute of New York University for their helps and exchange ideas on Network Coding and Opportunistic Routing implementation on OPNET.

VII. Conclusions

In this paper, we present NCOM, a network coding based overlay multicast design, that efficiently integrates Network Coding (NC), Opportunistic Routing (OR), and cross-layer link scheduling to achieve high efficiency and reliability multi-hop wireless multicast. In NCOM, we built up the minimum multicast overlay Steiner tree based on OR distance, which connecting source to all receivers. With random linear network coding is deployed, coded packets are sent multicastly along the overlay links toward receivers. NCOM fully exploit the broadcast nature of wireless transmission and the opportunistic packet reception cross adjacent overlay links. The transmissions of adjacent nodes in the overlay multicast are coordinated by a novel multicast acknowledgement MCCACK and cross-layer MAC scheduling. NCOM corrects some weaknesses of other multicast OR schemes on: multicast tree design, node coordination rely on offline link state measurements and lacking of cross-layer link scheduling.

NCOM is implemented on OPNET by customizing the IEEE 802.11b wireless LAN network modules. Through OPNET simulations, we demonstrate that NCOM can achieve higher throughput and lower source transmission redundancy than the existing NC and OR based wireless multicast designs. In our experiment, the average thoughput improvement of NCOM over Pacifier, the state of art multicast OR protocol, by approximately 63%. The average source redundancy is as small as 50% that value of Pacifier. NCOM get higher throughput gain and smaller packet latency due to its deployment of more forwarding nodes in the OR transmission progress.

As other future works, we will implement and evaluate performance of NCOM with other network coding techniques like [20] or [21]. We will also improve the performance of NCOM by investigating the optimal cross-layer scheduling for our multicast OR protocol.

References

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

[2] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a P2P Content Distribution system with Network Coding," in *IPTPS'06*, 2006.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *In Proc. ACM Sigcomm*, 2007.

[4] D. Koutsonikolas, Y. C. Hu, and C.-C. Wang, "Pacifier: High-throughput, reliable multicast without "crying babies" in wireless mesh networks." in *IEEE Infocom 2009*, Rio de Janeiro, Brazil, April 19-25, 2009.

[5] L. Li, R. Ramjee, M. Buddhikot, and S. Miller, "Network coding-based broadcast in mobile ad hoc networks," in *Proceedings of IEEE INFOCOM*, 2007, pp. 1739–1747.

[6] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," in *In Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, Nov. 2003.

[7] L. Xiao, M. Johansson, and S. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE Transactions on Communications*, vol. 52, no. 7, pp. 1136–1144, 2004.

[8] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *43rd IEEE Conference on Decision and Control*, 2004.

[9] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and mac for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, 2006.

[10] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Optimal cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proceedings of IEEE INFOCOM*, 2006.

[11] U. Akyol, M. Andrews, P. Gupta, J. D. Hobby, I. Saniee, and A. L. Stolyar, "Joint scheduling and congestion control in mobile ad-hoc networks," in *Proceedings of IEEE INFOCOM*, 2008.

[12] T. L. C. Tao Cui Ho, "Distributed minimum cost multicasting with lossless source coding and network coding," in *46th IEEE Conference on Decision and Control*, 2007, pp. 506 – 511.

[13] T. Cui, L. Chen, and T. Ho, "On distributed scheduling in wireless networks exploiting broadcast and network coding," *Trans. Comm.*, vol. 58, no. 4, pp. 1223–1234, 2010.

[14] Y. Liu, "Optimal Cross-layer Scheduling for Multicast in Multi-Channel Wireless Networks," *Technical Report, Polytechnic Institute of NYU*, December 2009, http://eeweb.poly.edu/faculty/yongliu/docs/multicast_tech.pdf.

[15] D. S. J. D. Couto, D. S. J. De, C. Daniel, R. Morris, D. Aguayo, and J. Bicket, "A high-throughput path metric for multi-hop wireless routing," in *Proc. of ACM MOBICOM*, 2003.

[16] H. Dubois-Ferrire, M.Grossglauser, and M. Vetterli, "Least-cost opportunistic routing," in *2007 Allerton Conference on Communication, Control, and Computing*, Monticello IL, September 2007.

[17] R.C.Shah, S.Wietholter, A.Wolisz, and J.M.Rabaey, "When does opportunistic routing make sense ?" in *in IEEE PerSens*, Mar.2005.

[18] K. Zeng, W. Lou, and H. Zhai, "On End-to-end Throughput of Opportunistic Routing in Multirate and Multihop Wireless Networks," in *IEEE Infocom 2008*, Phoenix, AZ, April 15-17, 2008.

[19] T.Le and Y.Liu, "On the Capacity of Hybrid Wireless Networks with Opportunistic Routing," in *WASA'09*, Boston, USA, August 2009.

[20] Y. Lin, B. Li, and B. Liang, "Codeor: opportunistic routing in wireless mesh networks with segmented network coding," in *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP)*, Orlando, Florida, USA, October 2008.

[21] Y. Lin, B. Liang, and B. Li, "Slideor: online opportunistic network coding in wireless mesh networks," in *INFOCOM'10: Proceedings of the 29th conference on Information communications*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 171–175.

[22] D. Koutsonikolas, C.-C. Wang, and Y. Hu, "Ccack: Efficient network coding based opportunistic routing through cumulative coded acknowledgments,," in *In Proc. 29th IEEE Conference on Computer Communications (INFOCOM)*, San Diego, CA, USA., March 2010.

[23] C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," in *in SIGCOMM Computer and Communications Review*, 2006.

[24] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: Reliable multicasting with an overlay network," in *Proceedings of Operating Systems Design and Implementation*, 2000, pp. 197–212.

[25] Y. Chu, S. Rao, and H. Zhang, “A case for end system multicast,” in *Proceedings of ACM SIGMETRICS*, 2000.

[26] R. Laufer, H. Dubois-Ferrire, and L. Kleinrock, “Multirate Anypath Routing in Wireless Mesh Networks,” in *IEEE Infocom 2009*, Rio de Janeiro, Brazil, April 2009.

[27] S. E. Dreyfus and R. A. Wagner, “The Steiner problem in graphs,” in *Networks*, 1972, pp. 195–207.

[28] B. Chazelle, “A minimum spanning tree algorithm with inverse-ackermann type complexity,” *J. ACM*, vol. 47, no. 6, pp. 1028–1047, 2000.

[29] C. Su and L. Tassiulas, “Mobile user’ s memory management to minimize deadline misses of users requests in a data broadcasting system,” in *Proceedings of 15th International Teletraffic Congress*, June 1997, pp. 223–232.