On the Robustness of Wireless Opportunistic Routing Toward Inaccurate Link-Level Measurements

Patrick P. C. Lee†, Vishal Misra‡, Dan Rubenstein‡
†The Chinese University of Hong Kong, Hong Kong ‡Columbia University, USA

***Abstract*—The quality of wireless links is inherently dynamic, and this often makes the measurements of link delivery probabilities inaccurate over short timescales. We present *Stable Opportunistic Routing (SOR)*, which improves unicast throughput for wireless mesh routing in the presence of inaccurate link-level measurements. In essence, SOR integrates two key features of prior approaches: (i) nodes trigger or suppress transmissions by inferring the actual reception of packets by neighboring nodes through channel overhearing, and (ii) nodes use network coding to avoid redundant transmissions. In addition, SOR is less dependent on accurate timing estimates or measured loss rates than prior approaches. Our stance is to argue that by carefully incorporating prior approaches into the design space of SOR, we can make opportunistic routing more robust toward link-level measurement errors, a practical issue in wireless mesh routing. Using nsclick simulation, we show that SOR has higher throughput than existing shortest-path and opportunistic routing protocols in large-scale networks, and the performance gain is more prominent when link-level measurements are erroneous.**

***Index Terms*—wireless mesh networks, opportunistic routing, performance evaluation**

I. Introduction

Empirical studies [1], [22], [24] show that wireless links are inherently *dynamic*, meaning that their delivery probabilities heavily fluctuate on millisecond timescales. Link quality can further be distorted by background interference and external sources (e.g., microwave [24]). The dynamic nature of wireless links complicates the design of wireless mesh routing protocols, especially when routing decisions are based on the assumption of stable channel losses over small timescales.

For instance, routing protocols such as shortest-path routing (e.g., ETX [6] and WCETT [7]) and opportunistic routing (e.g., ExOR [4] and MORE [5]) utilize periodic link-level probe measurements to infer existing channel conditions. Measurement results are periodically taken over large timescales (e.g., every 90s [6]) and distributed via link-state flooding to the entire network, so that all nodes can agree upon a routing decision that gives the highest possible throughput for data transfer. However, the overhead of link-state flooding reduces the frequency with which such probes can be issued. Thus, the estimates of link delivery probabilities are often *inaccurate* for the shorter timescales over which accurate estimates are needed; this inaccuracy leads to degraded throughput.

In this paper, we design a robust wireless mesh routing protocol called Stable Opportunistic Routing (SOR) that maintains high throughput in environments with dynamic channel loss conditions, which lead to inaccurate link-level measurements. In SOR, nodes trigger or suppress packet transmissions based on any packets they overhear being sent by neighboring nodes, and in addition, use network coding [2] to avoid transmitting redundant packets. Since the on-the-fly decision of transmitting packets is purely based on the actual reception of packets by neighboring nodes, SOR is less dependent on accurate timing estimates or measured loss rates as in prior approaches (see Section II). This makes SOR more resilient even though the measured link delivery probabilities deviate from the actual values. Also, SOR runs atop 802.11 MAC as a layer 2.5 protocol and enables us to deploy it in off-the-shelf wireless cards.

In essence, SOR combines the key features of ExOR and MORE: coordinating transmissions of nodes through channel overhearing as in ExOR, and randomly mixing packets using network coding as in MORE. While ExOR and MORE individually outperform shortest-path routing, ExOR is known to poorly exploit spatial reuse due to its stringent scheduling (see [5], [10]), and MORE is vulnerable to the deviation of link-level measurements due to its credit-based forwarding approach (see our analysis in Section III-B). We argue that by carefully combining their key components, SOR not only removes the limitations of ExOR and MORE, but also becomes fundamentally robust toward inaccurate link-level measurements. In summary, our contributions include:

- We analytically show that inaccurate estimates of link delivery probabilities can significantly degrade the throughput of current routing protocols.
- We address the design challenges of integrating ExOR and MORE into SOR.
- Using nsclick [23] simulation, we show that SOR outperforms MORE (opportunistic routing) and ETX (shortest-path routing) in various scenarios. For instance, in large-scale topologies with accurate measurements, SOR's throughput is up to $1.5\times$ and $2.1\times$ over MORE and ETX, respectively. Also, when the measured link delivery probabilities deviate from the actual values, SOR achieves, at best, $8.5\times$ and $2.9\times$ throughput gains over MORE and ETX, respectively.

The paper proceeds as follows. Section II reviews current opportunistic routing protocols. Section III motivates the need of robust mesh routing toward dynamic wireless links. Section IV describes the design of SOR. Section V reports results from nsclick simulation. Finally, Section VI concludes.

II. Related Work

Opportunistic routing has been proposed to improve throughput over shortest-path routing protocols such as ETX [6] and WCETT [7] by exploiting the opportunistic reception of forwarding nodes. This section presents the design limitations of existing opportunistic routing protocols, especially under the inherently varying channel conditions.

ExOR [4] is the first opportunistic routing implementation for wireless mesh networks. Its goal is to coordinate transmissions among multiple forwarders. A sender broadcasts a data packet to multiple forwarders, among which the receiver that is "closest" to the destination actually forwards the packet. To avoid redundant transmissions, ExOR imposes a strict timing order in which forwarders need to await predetermined packet-time estimates before accessing the channel. Accurate packet-time estimates are hard to determine, and the strict timing order prevents nodes from exploiting spatial reuse [5]. Extensions of ExOR include SOAR [25] and XCOR [16], but they still require forwarding timers to strictly schedule transmissions of nodes as in ExOR.

MORE [5] addresses ExOR's limitation of strictly scheduling node transmissions by using network coding [2]. Given a batch of packets, a source continuously generates randomly coded packets. Each forwarder, upon receiving a coded packet, is given some *credits*, which will be consumed for each packet to be forwarded. While MORE eliminates the overhead of node coordination as in ExOR, the allocation of credits is computed purely based on previously measured link delivery probabilities. If the link-level measurements are inaccurate or cannot adapt quickly enough to the current network condition, forwarders may receive too few (or too many) credits, leading to insufficient (or overloaded) transmissions and hence degraded throughput. Extensions of MORE include MC^2 [10] and CodeOR [20], yet they still use the previously measured link delivery probabilities to compute the expected number of packets to be forwarded.

III. Motivation

In this section, we motivate the need of a robust wireless mesh routing protocol toward inaccurate link-level measurements. Our major focus is to show via mathematical analysis that the measurement errors of link delivery probabilities can degrade existing routing protocols.

A. Testbed Measurements

We first demonstrate via testbed measurements that wireless links are dynamic and their link delivery probabilities fluctuate on millisecond timescales. Figure 1 depicts our mesh network testbed. The testbed comprises nine nodes, eight of which are located on the same floor, and node 8 is located one floor above the room where node 5 resides. Each mesh node is installed with an Atheros wireless card and the Madwifi driver [21]. We configure nodes to operate in the 802.11b ad-hoc mode using channel 1 (2.412GHz), transmission power 11dbm, and the fixed bit rate 11Mbps.

Each node *in turn* broadcasts 1.4-KB packets as fast as possible for 120s, and we collect the statistics at all other receiver nodes before another node broadcasts packets. The 1.4-KB size is close to that of the data packets when we evaluate the routing protocols (see Section V). Hence, our test

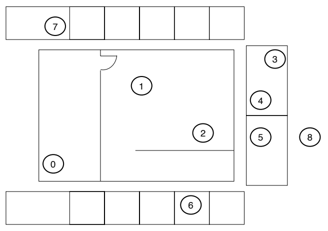


Fig. 1. Our wireless mesh testbed.

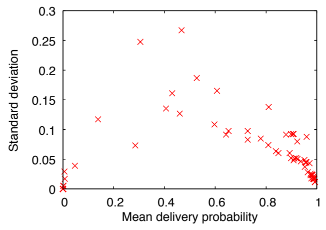


Fig. 2. Means versus standard deviations of link delivery probabilities.

serves as an indicator for the instantaneous behavior when the routing protocols transfer data.

Figure 2 shows the scatter plot of the means and standard deviations of the link delivery probabilities of all 72 links in our testbed. Each data point corresponds to a link and is computed from the 200-ms samples over the 120-s span. We see that links with intermediate delivery probabilities experience higher standard deviations. Note that links with intermediate delivery probabilities are not uncommon, for example, as shown in Roofnet experiments [1]. We also plot the link delivery probabilities versus time for some specific links in Figure 3. We observe that the higher-loss links $0 \rightarrow 4$ and $8 \rightarrow 1$ (respectively with average delivery probabilities 0.56 and 0.16) have more fluctuating delivery probabilities when compared to the lower-loss link $1 \rightarrow 3$ (with average delivery probability 0.99). Also, the delivery probability of link $0 \rightarrow 4$ occasionally drops to zero.

We observe the fluctuations of link delivery probabilities on small timescales, and the results conform to the measurements in other testbeds [1], [22], [24]. In addition, [24] shows that link delivery probabilities can be significantly reduced by nearby energy sources such as microwave.

Does rate adaptation help? To handle the variance of link delivery probabilities, one possible solution is *rate adaptation*, such that nodes with dynamic links switch to lower transmission rates for more stable link quality. However, deploying rate adaption poses some open design issues. First, opportunistic routing protocols such as ExOR and MORE use the broadcast mode for transmissions and disable link-layer ARQs, which are needed by today's rate adaptation implementation for link quality inference. In addition, a node transmitting at a low rate can hurt the throughput of neighboring nodes that transmit at high rates (i.e., the performance anomaly problem [11]), and the overall throughput might not improve. While exploring the advantages of rate adaptation in opportunistic routing is important, it is still essential for a routing protocol itself to be robust toward link quality fluctuations.

B. Mathematical Analysis

Given the link quality fluctuations on small timescales, the estimates of link delivery probabilities may become inaccurate and deviate from the actual delivery probabilities by the time data forwarding begins. We now show via mathematical analysis how the shortest-path routing protocol ETX and the opportunistic routing protocol MORE are sensitive to the

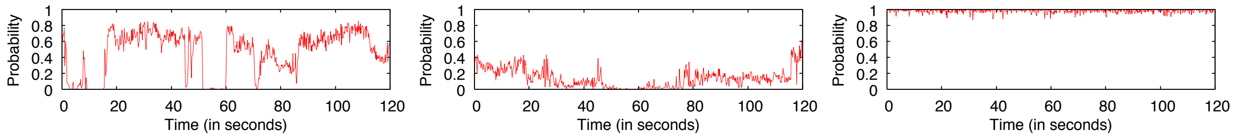
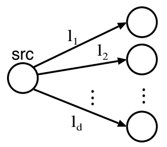
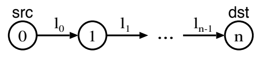


Fig. 3. Delivery probability versus time for different links. Link $0 \rightarrow 4$ (left) and link $8 \rightarrow 1$ (middle) are more dynamic than link $1 \rightarrow 3$ (right).



(a) Section III-B1



(b) Section III-B2

Fig. 4. Topologies in Mathematical Analysis.

measurements of link delivery probabilities and have their throughput degraded.

Let p_l and $p_l + \epsilon_l$ be the measured and actual link delivery probabilities of link l, and hence ϵ_l denotes the error in measurements. We assume that packet losses are independent. Due to the probabilistic nature of a link, the actual number of transmissions to deliver a packet across a link varies for different packets. Thus, we assume that our results are derived in the expected sense, i.e., a node *always* makes the expected number of transmissions to deliver a packet across a link.

To demonstrate our major results, we consider two particular loss models as motivating examples. We also resort to simulation in Section V to evaluate more general settings.

1) Impact of Number of Forwarders when $\epsilon_l \sim U(-e, e)$: We first show how the number of transmissions needed to deliver a packet across a link varies with the number of forwarders (see Figure 4(a)). We assume that ϵ_l follows a uniform distribution $U(-e, e)$ for some $e > 0$. This captures the scenario where link delivery probabilities, which are sampled from periodic probing, contain statistical errors.

Note that MORE delivers each packet to multiple forwarders (say, across d links $l_1, l_2, \cdots, l_d$), and ETX delivers each packet to a single forwarder (say, across link l_1) (see Figure 4(a)). The number of transmissions required by MORE to deliver a packet to at least one of the d forwarders is

$$T(d) = \left[1 - \prod_{i=1}^{d}(1 - (p_{l_i} + \epsilon_{l_i}))\right]^{-1},$$

where ETX arises as a special case with $d = 1$.

We now use numerical simulation to see how $T(d)$ varies when $\epsilon_l \sim U(-e, e)$ for some $e > 0$. We fix $p_{l_i} = 0.6$ for all l_i. Then for the given d and e, we randomly generate 1,000 sets of ϵ_{l_i}'s and compute the average of $E[T(d)]$. Figure 5 plots $T(d)$ versus e. We note that $E[T(1)]$ increases with e, even though ϵ_l has a zero mean. To explain this counter-intuitive observation, we can in fact show that

$$E[T(1)] = E[\frac{1}{p_{l_1} + \epsilon_{l_1}}] = \frac{1}{2e}\log\frac{p_{l_1} + e}{p_{l_1} - e},$$

which is an increasing function of e. This means ETX's throughput decreases with higher deviations of measurement errors, even though the errors have a zero mean. However, the decrease in throughput is mitigated with multiple forwarders (i.e., with a higher value of d), as in opportunistic routing.

2) Impact of Path Length when $\epsilon_l < 0$: We now show how the number of transmissions needed to deliver a packet from source to destination varies with the path length (see Figure 4(b)). We assume that $\epsilon_l < 0$ for all l. This captures the case where a network suffers from background interference and the actual link delivery probabilities are less than the measured values.

We focus on MORE, whose number of transmissions of a node i depends on its allocated credits (denoted by C_i). Let T_i be the expected number of transmissions made by node i to deliver a packet to the destination, and R_i be the expected number of packets that node i needs to receive so as to make T_i transmissions. Note that C_i, R_i, and T_i are related as follows:

$$T_i = R_i \times C_i.$$

Consider a line topology shown in Figure 4(b), in which there is a single flow with source 0 and destination n, and node i forwards packets to node $i+1$ through link l_i. In general, MORE uses multiple forwarders. In this case, each node in Figure 4(b) denotes an aggregate of multiple forwarders, and the link delivery probability is the probability to deliver a packet to at least one forwarder. Since p_{l_i} is the measured link delivery probability of link l_i, one can show that node i has credits $C_i = 1/p_{l_i}$ for all i.

We now analyze how T_0 varies with the path length n, given that $\epsilon_l < 0$. We compute T_i in descending order of i. For node $n-1$ to deliver a packet to destination n, it needs $T_{n-1} = (p_{l_{n-1}} + \epsilon_{l_{n-1}})^{-1}$ transmissions. To make T_{n-1} transmissions, node $n-1$ needs to receive $R_{n-1} = T_{n-1}/C_{n-1} = p_{l_{n-1}}/(p_{l_{n-1}} + \epsilon_{l_{n-1}})$ packets. This requires node $n-2$, the upstream node of node $n-1$, to transmit $T_{n-2} = R_{n-1}/(p_{l_{n-2}} + \epsilon_{l_{n-2}}) = p_{l_{n-1}}/[(p_{l_{n-2}} + \epsilon_{l_{n-2}})(p_{l_{n-1}} + \epsilon_{l_{n-1}})]$ packets. By induction, the expected number of transmissions of source 0 is

$$T_0 = \frac{\prod_{i=1}^{n-1} p_{l_i}}{\prod_{i=0}^{n-1}(p_{l_i} + \epsilon_{l_i})}.$$

Figure 6 shows T_0 versus e for different values of n, where we fix $p_{l_i} = 0.8$ and $\epsilon_{l_i} = -e$ for some $e > 0$. When $\epsilon_l < 0$ for all l, the number of transmissions by source 0 to deliver a packet to the destination *increases exponentially with path length n*. The intuitive reason is that MORE is built upon *end-to-end recovery*, as the source needs to push forward enough credits so as to recover the packet loss in any intermediate link. Thus, although MORE eliminates node coordination (i.e., no channel overhearing is needed), its credit-based forwarding approach is inherently sensitive to the slight degradations of link delivery probabilities, and this leads to a dramatic decrease in throughput. We use simulation to confirm this limitation for general networks in Section V.

To make MORE, or opportunistic routing in general, robust toward inaccurate link-level measurements, we propose to use

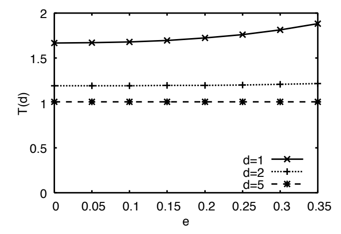


Fig. 5. $T(d)$ vs. e (Sec. III-B1).

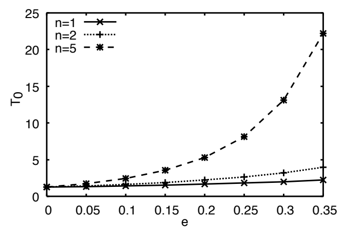


Fig. 6. T_0 vs. e (Sec. III-B2).

hop-by-hop recovery, in which intermediate nodes can trigger retransmissions if they infer that their forwarded packets do not reach their downstream neighbors. Hop-by-hop recovery is also used in ExOR, yet ExOR relies on accurate time estimates to coordinate node transmissions, thereby reducing spatial reuse. Thus, it is important to carefully implement node coordination (as in ExOR) while maximizing spatial reuse (as in MORE). Section IV addresses this key design challenge.

IV. SOR Design

Current shortest-path and opportunistic routing protocols make routing decisions based on the link delivery probabilities obtained from the probe measurements over large timescales (e.g., every 90s [6]). However, as shown in Section III, the measurements may deviate from the actual delivery probabilities over short timescales (e.g., in milliseconds). In this section, we propose SOR, an opportunistic routing protocol that seeks to be robust toward inaccurate link-level measurements. In a nutshell, SOR nodes infer the actual reception of packets by neighboring nodes (as in ExOR), and dynamically trigger transmissions if non-redundant packets can be forwarded, or suppress transmissions otherwise. Also, SOR uses network coding to further avoid redundant transmissions (as in MORE).

A. Assumptions

The assumptions of SOR are built upon ExOR and MORE. We assume that SOR is deployed under centralized administration, where nodes are identified by unique node IDs.

SOR transmits packets on a *per-batch* basis, such that each batch contains a fixed number of packets and SOR attempts reliable delivery for all batches. Thus, we assume that SOR is only used by *bulk data transfer applications* so that it can accumulate packets into batches. Like ExOR and MORE, which are batch-based, we do not expect SOR to be used in interactive or short data transfer applications. As addressed by ExOR's authors [3], opportunistic routing is also expected to perform poorly when being layered under TCP due to packet reordering and TCP's rate control mechanisms.

Since we only consider bulk data transfer, our objective is to improve end-to-end throughput. Other performance metrics such as transfer delay are not our emphasis.

SOR's benefits come from the assumption that nodes can infer the actual reception of packets by neighboring nodes. We assume that the network is dense enough so that each node is within the transmission range of multiple neighboring nodes and can overhear their transmissions. Having a dense network is also necessary for opportunistic routing schemes like ExOR and MORE, so that nodes can exploit opportunistic transmissions along multiple routes.

We also assume that link delivery probabilities are independent, which is a valid approximation in practice [1], [24].

B. Design Intuition of SOR

SOR uses randomized network coding [12], i.e., each sender node selects uniformly at random a set of code coefficients over a finite field and forms a linear combination of the current batch of data packets that are composed of either the *native* (i.e., original) packets (for the source) or the encoded packets that have thus far been received (for forwarders). The destination then decodes the received coded packets to recover original data using Gaussian elimination.

As shown in [12], the probability that these randomly coded packets are *innovative* (i.e., linearly independent) is very high. Suppose that the source (the originator of native packets) generates K randomly encoded packets from a batch, where K is equal to or slightly larger than the batch size. Upon receiving innovative packets, each node should re-encode them and transmit new innovative packets. It stops transmissions and frees the channel until its transmitted packets are received by nodes "closer" to the destination. When the destination receives K innovative packets, it can recover the original data with a very high probability.

To know when to trigger and suppress transmissions, a SOR node needs to infer what packets its neighboring nodes actually receive by overhearing their forwarded packets. However, inferring the reception of any encoded packet is not straightforward, since when a forwarder forwards any packet, this forwarded packet is randomly mixed with the packets received by the forwarder. By looking at the payload or the code coefficients stored in the forwarded packet, one cannot tell which native packets contribute to the forwarded packet. This implies that the forwarded packet needs to include some indicative information to tell a node whether its transmitted packets have been correctly received by its neighbors. In view of this, we propose to associate each forwarded packet with a *packet sequence number (PSN)*. In traditional routing where no coding is used, a sequence number serves as a unique identifier of a packet. However, in SOR where network coding is used, a PSN has a different definition, i.e., a PSN specifies one of the native packets that contributes to the encoded packet. We will later explain the assignment of PSNs in more detail.

To understand SOR, we analyze how a node processes a single batch of data packets so as to have the batch reliably delivered to the destination. Before forwarding any data, all nodes agree on the set of forwarders (Section IV-C). When the source has a new batch of data packets, it starts transmissions, while all other nodes await packets from upstream neighbors. Upon receiving enough packets, a node starts forwarding the batch (Section IV-D). It triggers or suppresses its transmissions for the batch based on the PSNs tagged in the packets being overheard (Section IV-E). A node stops transmitting the current batch when it is acknowledged (Section IV-F).

C. Preparing for Transmissions

1) Link-level Measurements: Each SOR node periodically measures link delivery probabilities via periodic link-level probes. In actual deployment of SOR, link-state information is distributed to all nodes through periodic link-state flooding. However, link-state flooding incurs communication overhead, which also exists in current routing protocols (e.g., [4], [5], [6], [7]). Thus, without hurting the fairness of our routing protocol comparisons, we neglect this overhead in our evaluation. Instead, we assume that the link-state information is available to all nodes before they forward data.

The link-level estimates assist SOR to make the best possible guess of the forwarders that can bring packets "closer" to the destination (see Section IV-C2) and to initiate the transmission of a batch (see Section IV-D). However, the actual link delivery probabilities vary over short timescales. Thus, nodes must decide on the fly when to trigger or suppress transmissions of individual packets based on the actual reception of neighboring nodes, and this is the key motivation of SOR.

2) Forwarder Selection: Like ExOR and MORE, we assume that for a given destination, the forwarding priorities of all nodes are prioritized from highest to lowest based on the ascending order of the ETX metric [6] (the expected number of transmissions to deliver a packet to the destination). Higher-priority nodes are the more preferred forwarders. We can also replace ETX with other opportunistic-routing metrics (e.g., [8], [29]) for a higher throughput gain, yet we choose ETX as a base case for our evaluation. To avoid medium contention of too many forwarders, we assume that each sender has at most k forwarders. Here, we set $k = 8$.

3) Packet Header: SOR has two types of packets: data and ACK. A data packet contains the encoded data, while an ACK packet is for the destination to acknowledge the source the receipt of the current batch of data packets (see Section IV-F). For each data packet, a packet header (shown in Figure 7) is included between the MAC header and the encoded data payload. The Type field distinguishes data and ACK packets. Src and Dst are the node IDs of the source and the destination of a flow, respectively, and the Sender field is the node ID of the current sender of the data packet. BatchID identifies the batch that the data packet belongs to. ACKMap is a bitmap that acknowledges lower-priority nodes which PSNs have been received (see Section IV-E). CodeVector is the list of coefficients encoded for a data packet. NumFwder specifies the number of forwarders specified in the ForwarderList. Suppose that the network contains at most 256 nodes and that we apply network coding on a batch size 32 over the Galois Field GF(2^8). Then we can compact each node ID and each code coefficient into a single byte. This makes the resulting header size at most 55 bytes, which is comparable to the 70-byte header size used in MORE [5]. For 1.4-KB payload, both SOR and MORE have no more than 5% header overhead, which we expect has limited impact on the resulting throughput. On the other hand, an ACK packet contains only the fields Type, Src, Dst, Sender, and BatchID, and has zero data payload.

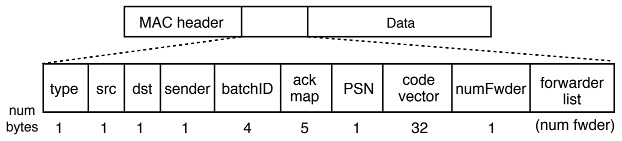


Fig. 7. Packet header for a data packet in SOR.

D. Initiating the Transmission of a Batch

When a forwarder receives packets from a sender, it needs to decide when it should start forwarding packets. If the forwarder starts too early, it cannot generate too many innovative packets from the received encoded packets, while it contends the channel with the sender that is still transmitting packets. On the other hand, if the forwarder starts too late, the sender may over-generate encoded packets, among which the latter ones are non-innovative. This is in essence the *spatial pipelining problem* [19], in which forwarder nodes need to start transmissions at the right time for effective spatial reuse.

We borrow the idea from MORE [5], and propose a heuristic in which a forwarder starts forwarding packets as soon as it has received the expected number of packets within a batch that will reach the forwarder itself. Without loss of generality, we consider a set of nodes $1, 2, \cdots$ ordered in descending order ETX values for a given destination (i.e., node j is "closer" to the destination than node i for $i < j$). Let p_{ij} be the link delivery probability from node i to node j, and $F(i)$ be the set of forwarders selected in advance by node i. We now consider L_j, denoting the expected number of packets that node j is responsible for forwarding. Note that L_s equals the batch size for source s, and $L_t = 0$ for destination t. To compute L_j, we first compute the proportion of packets that node $j \in F(i)$ will forward for node i. Such packets are those that have been received by node j but not by the higher-priority forwarders of node i. Hence,

$$L_j = \sum_{i=1, j \in F(i)}^{j-1} L_i \frac{p_{ij} \prod_{k \in F(i), k > j} (1 - p_{ik})}{1 - \prod_{k \in F(i)} (1 - p_{ik})}.$$

L_j can be readily computed in ascending order of j [5]. Thus, the expected number of packets (denoted by R_j) that can reach node j for a given batch of data packets is:

$$R_j = \sum_{i=1}^{j-1} L_i p_{ij}.$$

Note that L_j and R_j can be distributed to all nodes in the network via link-state flooding (see Section IV-C).

MORE [5] also uses L_j and R_j compute credits that specify the number of transmissions that a node makes for every packet received. Our design is fundamentally different from MORE in that we use L_j and R_j to initiate the *first* transmission of a batch of data packets for the purpose of spatial pipelining, instead of triggering the transmissions of individual packets as in MORE. In SOR, once a node is initiated for transmitting a batch, it will transmit packets as long as it has innovative packets needed by its higher-priority forwarders, and the decision is based on the actual reception of those forwarders (see Section IV-E).

E. Triggering/Suppressing Transmission of a Batch

To decide whether to transmit data packets in a batch, each SOR node independently associates the batch with two local arrays *LowMap* and *HighMap*, which track the PSNs of the packets that have been received by its lower-priority (upstream) and higher-priority (downstream) neighboring nodes, respectively. Initially, all elements of both LowMap and HighMap are set to zero, except for the source (i.e., the originator of packets), whose elements of LowMap are all set to one.

Intuitively, the PSN tagged in an encoded data packet specifies one of the native packets within a batch that contributes to the encoded data packet. Our goal is to deliver to the destination a batch-size number of encoded data packets tagged with distinct PSNs, so that the destination can decode and recover the batch of native data packets with a very high probability (see Section IV-B).

A node triggers its transmission of a data packet as long as there exists i such that LowMap[i] == 1 and HighMap[i] == 0 (i.e., i is the PSN that has been received by lower-priority neighbors but not yet received by higher-priority neighbors). The value of such i will then be set in the PSN field of the encoded data packet to be forwarded. If no such i exists, it means that the forwarded packets will not help higher-priority nodes generate innovative packets, so the node will suppress its transmissions. We cycle through LowMap and HighMap for the next available i to set the PSN field. In addition, the ACKMap field of the forwarded data packet is set to LowMap held by the sender node, such that the ith bit of ACKMap is set if LowMap[i] == 1.

When a node receives a data packet (call it packet x), it compares its forwarding priority with that of the sender node of x. If the receiver has a higher priority and is a forwarder selected by the sender, then it sets LowMap[i] = 1, where i is the PSN tagged in x. This means that the ith native packet is one of the native packets used by the lower-priority sender to construct x. On the other hand, if the receiver has a lower priority, then it sets HighMap[i] = 1 if the ith bit of ACKMap in x is set, meaning that the higher-priority sender is using the ith native packet to encode forwarded packets.

Since the destination does not forward packets, we have it broadcast zero-payload data packet with its current LowMap upon receiving new PSNs.

To further increase the successful decoding probability, we set the array sizes of LowMap and HighMap to be 10% larger than the batch size. This allows the destination to receive 10% more differently encoded packets than the batch size, and hence the probability that these packets can be successfully decoded will be much more close to one.

The use of the ACKMap field in SOR is similar to the *batch map* field in ExOR [4]. In ExOR, each forwarded packet has a batch map field that specifies the *exact* packet copies that have been received by higher-priority nodes, while the ACKMap field in SOR specifies which native packets contribute to the encoded packets received by higher-priority nodes.

F. Acknowledging Receipt of a Batch

The destination inserts any received data packet to the current batch of received data, and tries to decode the batch. If the destination successfully decodes the current batch, it sends an ACK packet to the source using shortest path routing (e.g., ETX) to indicate successful end-to-end delivery. We have all nodes enabled promiscuous mode, so that they can suppress the transmission of the current batch when they overhear the ACK packet or receive data packets from a newer batch.

G. Summary

In summary, SOR forwards useful information across the network by encoding received packets, without imposing a strict timing order of transmissions of nodes as in ExOR. Also, SOR nodes embed the information of encoded packets into the ACKMap field that allows neighbors to infer whether to trigger or suppress the transmissions of innovative and redundant packets, respectively. This avoids using loss-rate-based credits to control the sending rates of nodes as in MORE.

It is important to note that even with network coding, SOR does not totally eliminate the transmissions of non-innovative packets given a lossy, dynamic wireless medium. The overhead of non-innovative transmissions is reflected in the resulting throughput, which we will extensively study in Section V.

H. Other Implementation Issues

Handling Hidden Terminals. Like all opportunistic routing protocols, SOR transmits data packets in broadcast. Since no control frames are used in broadcast, the transmissions of data packets only use physical carrier sensing to avoid collisions due to hidden terminals and this could be ineffective. We propose a heuristic based on *pseudo-broadcast* [14], such that each SOR node switches to unicast when it has reused a PSN for more than m times for a given batch, where m is a tunable parameter. This allows nodes to utilize the 802.11 backoff mechanism for collision avoidance. As long as nodes have promiscuous mode enabled, they can still overhear packet transmissions of neighboring nodes. The trade-off is that the 802.11 scheduler will retransmit unacknowledged packets without re-encoding them, and hence we lose the gain of network coding. We set $m = 5$ in our evaluation.

Handling Concurrent Flows. To efficiently allocate channel bandwidth across multiple flows, we can modify the wireless driver and add congestion control in the MAC layer (layer 2) to regulate the sending rates of multiple flows coming from the SOR layer (layer 2.5). For instance, DiffQ [28] is a layer-2 congestion control implementation using a backpressure algorithm. Our current SOR implementation does not include any congestion control, and we evaluate how multiple flows react with this baseline implementation in Section V.

V. Simulation

We now evaluate different routing protocols in realistic wireless scenarios using *nsclick* [23] with the Madwifi extension [18]. The nsclick simulator embeds the Click modular router architecture [15] into the ns2 simulator [27], such that the routing protocols are developed with Click, while the physical (wireless) medium is simulated using ns2. The aim of using nsclick is that with only changes of configuration scripts, we can readily deploy our Click-based routing protocols on a real wireless mesh network.

Our simulation evaluates the throughput of three routing protocols: (i) SOR, (ii) MORE [5], which is downloaded from its authors' website and slightly modified to fit into nsclick, and (iii) ETX, a shortest-path routing protocol [6]. We do not consider ExOR [4], mainly because its source code is not published and its performance is shown to be worse than MORE [5]. In addition, since TCP does not interact well with opportunistic routing (see Section IV-A), our evaluation uses UDP as the transport layer for bulk data transfer.

Our simulation is based on a lognormal shadowing propagation model [26]. Let d_{ij} and p_{ij} be the distance and the delivery probability for the link from node i to node j, respectively. According to [26], p_{ij} can be approximated as a function of d_{ij} as follows:

$$p_{ij} = \begin{cases} 1 - ((\frac{d_{ij}}{R})^{2\beta})/2 & \text{if } d_{ij} \leq R \\ ((\frac{2R-d_{ij}}{R})^{2\beta})/2 & \text{if } R < d_{ij} \leq 2R \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where β is the power attenuation factor ranging from 2 to 6, and R is defined as the distance such that $p_{ij}(R) = 0.5$. Note that we also consider the case where $\beta = 0.5$, which means that p_{ij} is a linear function of d_{ij}. This linearity relationship is shown to be a good approximation based on a sensor-network testbed experiment [3], [9].

Our evaluation explores the impact of inaccurate link-level measurements on routing protocols. Let p_{ij} and $p_{ij}+\epsilon_{ij}$ denote the measured and actual delivery probabilities of the link from node i to node j, respectively, where p_{ij} is determined by Equation (1), and ϵ_{ij} denotes the measurement error. The actual link delivery probability $p_{ij} + \epsilon_{ij}$ specifies the actual likelihood of propagation loss of a data packet traversing the link from node i to node j. Our evaluation considers different distributions of ϵ_{ij}.

A data packet can be lost due to (i) the propagation loss as specified by the actual link delivery probability $p_{ij} + \epsilon_{ij}$, and (ii) the packet collisions as simulated by ns2. On the other hand, each of the routing protocols replies a control packet to indicate the receipt of a packet (i.e., MAC-layer ACK for ETX) or a batch (i.e., ACK for MORE and SOR). These control packets have much smaller sizes than the data packets, and hence have higher delivery probabilities [6]. Therefore, we assume that the control packets experience no propagation loss and are subject to loss only due to packet collisions.

Our simulation environment in ns2 is based on the 802.11b standard and the transmission range and the carrier-sensing range are the default values 250m and 550m, respectively. We set R = 125m, as $p_{ij} = 0$ for $d_{ij} \geq 2R$. All routing protocols operate under the fixed bit-rate 11Mb/s. We fix the payload size at 1.4KB, while the size of the packet header depends on the routing protocol, and fix the batch size at 32 packets for SOR and MORE. Also, we have the RTS/CTS handshake disabled. In each simulation, the source transmits data to the destination as fast as possible for 30s. We simulate each setting of parameters for 30 runs, and plot the average throughput and its 95% confidence interval.

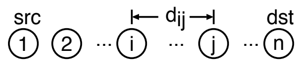
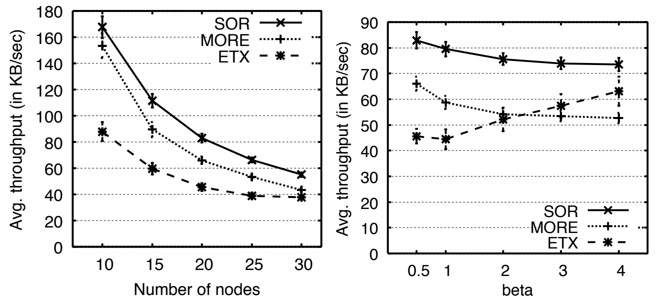


Fig. 8. Single-flow, line topology.



(a) Number of nodes (b) β

Fig. 9. Experiment 1: Throughput in a single-flow, line topology.

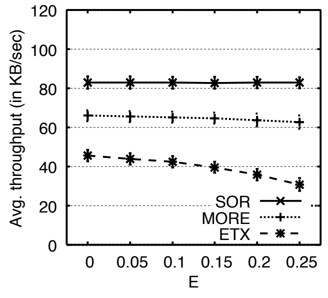
Experiment 1 (Single-flow, line topology with accurate link-level estimates). We start with a single-flow, line topology in Figure 8 and evaluate the throughput as a function of the path length of a flow. The topology contains n nodes. A single flow transmits data from node 1 to node n. We configure the distance between each pair of neighboring nodes i and $i+1$ to be uniformly distributed between 25m and 75m (i.e., $d_{i,i+1} \sim U(25, 75)$). We consider the case where all nodes have accurate link delivery probabilities (i.e., $\epsilon_{ij} = 0$).

Figure 9(a) shows the throughput versus different numbers of nodes, with $\beta = 0.5$. SOR's throughput is higher than both MORE and ETX by up to 30% and 90%, respectively. Intuitively, SOR outperforms MORE even with accurate link-level measurements because MORE nodes always forward a fixed number of packets for each received packet given by their transmit credits, while SOR nodes forward a varying number of packets based on the actual reception of packets by neighboring nodes through channel overhearing.

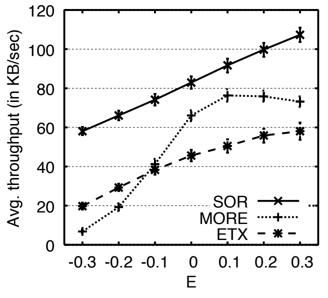
Figure 9(b) shows the impact of β on the throughput when we fix the number of nodes to be 20. As β increases, short-distance links become more reliable, while long-distance links become less reliable. Thus, ETX prefers to choose short links for its best route, and has improved throughput with increased β. On the other hand, when β increases, both MORE and SOR have fewer long-distance links for opportunistic forwarding and hence see throughput drops. Nevertheless, SOR remains to have higher throughput than ETX with all β, for example, by 80% when $\beta = 0.5$ and by 20% when $\beta = 4$. In the following experiments, we focus on $\beta = 0.5$, while we obtain similar results for $\beta = 2$.

Experiment 2 (Impact of inaccurate link-level measurements). When a flow starts transmitting packets, the actual link delivery probabilities could deviate from prior measurements. Here, we study such discrepancies, and validate our sensitivity analysis in Section III-B. We again focus on the single-flow, line topology in Figure 8, and we fix $n = 20$. As in Experiment 1, we set $d_{i,i+1} \sim U(25, 75)$, and fix $\beta = 0.5$.

Figure 10(a) first shows the throughput for the two-sided error case where $\epsilon_{ij} \sim U(-E, E)$ for $E \geq 0$. ETX's through-

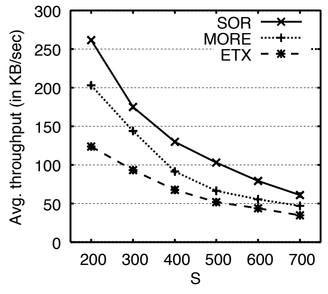


(a) Two-sided error

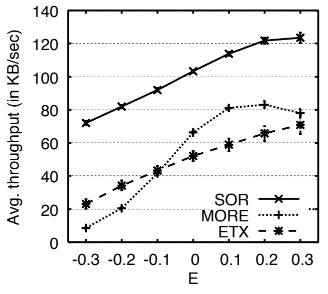


(b) One-sided error

Fig. 10. Experiment 2: Throughput versus inaccuracy ϵ_{ij}.



(a) $\beta = 0.5$, no error



(b) $\beta = 0.5$, one-sided error

Fig. 11. Experiment 3: General topology.

put decreases by 30% as E increases to 0.25, even though ϵ_{ij} has zero mean. In contrast, both MORE and SOR have stable throughput regardless of E, while SOR's throughput remains higher than MORE's. This also conforms to our analysis in Section III-B1.

Figure 10(b) shows the one-sided error case where ϵ_{ij} is uniformly distributed between 0 and E (i.e., $\epsilon_{ij} \sim U(E, 0)$ if $E < 0$, or $\epsilon_{ij} \sim U(0, E)$ if $E \geq 0$). When $E = -0.3$, SOR's throughput is 8.5× and 2.9× over MORE and ETX, respectively. When $E < 0$, the network is actually more lossy, but MORE nodes are not assigned enough credits to push forward packets to the destination, and hence the overall throughput significantly degrades. We also confirm this in Section III-B2.

When $E > 0$, the network is actually more reliable, so all protocols see increased throughput. However, MORE nodes are given excess credits to forward additional packets. This leads to more collisions, and hence the throughput drops when E further increases.

Experiment 3 (General topology). We now extend our analysis to a general topology, in which we place 100 nodes over a square plane of size S. We assume that there is a single flow, whose source and destination are placed at the bottom-left and top-right corners of the plane, respectively. The remaining nodes are randomly placed over the plane.

Figures 11(a) depicts the throughput versus S when $\beta = 0.5$, assuming that link-level measurements are accurate (i.e., $\epsilon_{ij} = 0$). As S increases, the network becomes sparser, and this degrades the link quality and throughput of all protocols. Nevertheless, SOR's throughput is 1.2× to 1.5× over MORE, and 1.7× to 2.1× over ETX.

Figures 11(b) shows the throughput with inaccurate link-level measurements, where ϵ_{ij} is the one-sided error uniformly distributed between 0 and E, and S is fixed to be 500. When $E < 0$, SOR's throughput is 2.2× to 8.5× over MORE, and 2.1× to 3.1× over ETX.

Experiment 4 (Multiple flows). We now study the robustness of SOR when there are multiple concurrent flows that interfere with each other. We focus on two metrics: (i) per-flow throughput and (ii) fairness. To measure fairness, we use Jain's *fairness index* [13], defined as $(\sum x_i)^2/(F \sum x_i^2)$, which accounts for F flows with throughputs $x_1, x_2, \cdots x_F$.

We start with an $r \times c$ lattice topology shown in Figure 12(a), in which each row corresponds to a single flow and each node is only responsible for the flow along the row. Let d be the distance between the neighboring nodes and c be the number of nodes in a row (or flow). In the experiment, we fix $d = 50$, $c = 8$, and $\beta = 0.5$.

Figures 13(a) and 13(b) show the per-flow throughput versus the number of flows (i.e., r), where we assume inaccurate link-level measurements. As expected, the per-flow throughput decreases when more interfering flows concurrently transmit data. Nevertheless, SOR's throughput is 1.1× and 1.4× over MORE, and up to 2.9× and 3.2× over ETX, when $\epsilon_{ij} \sim U(-0.2, 0.2)$ and $\epsilon_{ij} \sim U(-0.2, 0)$, respectively. Also, Figures 13(c) and 13(d) show the fairness versus r. Note that SOR maintains a high fairness index (≥ 0.98) even though the link-level measurements are inaccurate.

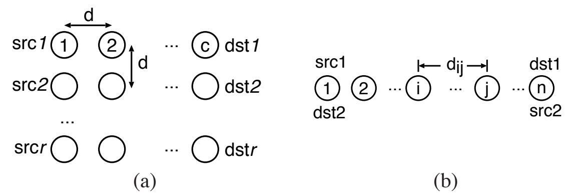
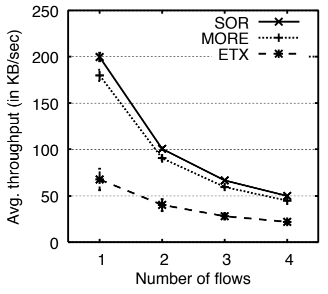
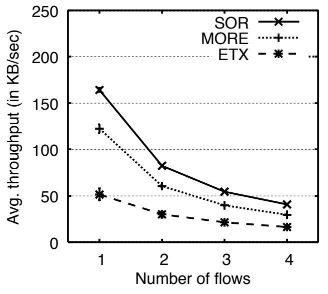


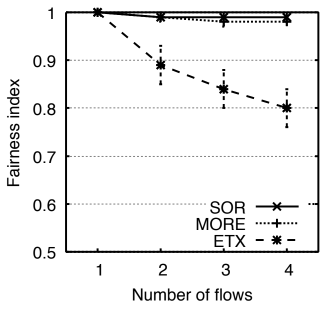
Fig. 12. Topologies for multi-flow experiment.



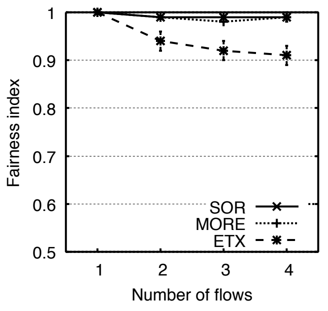
(a) Throughput, $\epsilon_{ij} \sim U(-0.2, 0.2)$



(b) Throughput, $\epsilon_{ij} \sim U(-0.2, 0)$

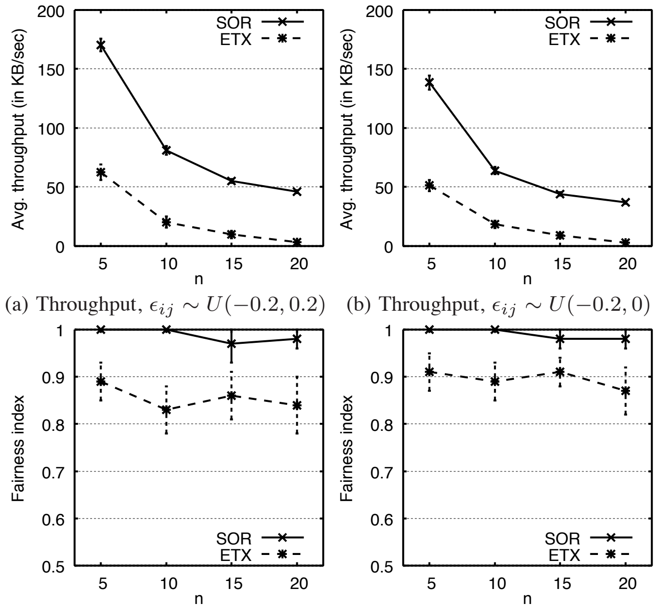


(c) Fairness, $\epsilon_{ij} \sim U(-0.2, 0.2)$



(d) Fairness, $\epsilon_{ij} \sim U(-0.2, 0)$

Fig. 13. Experiment 4: Multiple flows (for topology in Figure 12(a)).



(a) Throughput, $\epsilon_{ij} \sim U(-0.2, 0.2)$ (b) Throughput, $\epsilon_{ij} \sim U(-0.2, 0)$

(c) Fairness, $\epsilon_{ij} \sim U(-0.2, 0.2)$ (d) Fairness, $\epsilon_{ij} \sim U(-0.2, 0)$

Fig. 14. Experiment 4: Multiple flows (for topology in Figure 12(b)).

To see how a node simultaneously handles multiple flows, we switch to a line topology in Figure 12(b), which contains two flows from 1 to n and from n to 1. We let $d_{ij} \sim U(25, 75)$ and $\beta = 0.5$. In the currently published implementation of MORE, each node supports only a single flow. Thus, we focus on SOR and ETX only. Specifically, we do not implement any congestion control algorithm: In SOR, each node inspects flows in a round-robin manner and forwards any encoded packet when available; in ETX, each node simply forwards any received packet. We evaluate these baseline implementations of SOR and ETX (see discussion in Section IV-H).

Figures 14(a) and 14(b) show the per-flow throughput versus n, the length of the line topology. SOR's throughput is at least $8\times$ over ETX when $n = 20$. In addition, SOR still keeps a high fairness index (≥ 0.98) for all n.

Summary: We show that for various topologies, SOR maintains higher throughput than MORE and ETX when the actual link delivery probabilities deviate from the measured values. Also, SOR remains robust with multiple concurrent flows, in terms of per-flow throughput and fairness.

VI. Conclusions

It is important for wireless mesh routing to maintain high throughput in inherently dynamic wireless networks that make the measured link delivery probabilities deviate from the actual values. As analytically shown in this paper, the inaccurate measurements can significantly degrade the throughput of existing mesh routing protocols. This motivates us to design SOR. While each of the design blocks of SOR is built upon existing opportunistic routing schemes, we show via extensive simulation that their combination enables SOR to improve throughput in various settings, such as general topologies and multi-flow scenarios, where we have inaccurate estimates of link delivery probabilities.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *Proc. of ACM SIGCOMM*, Aug 2004.

[2] R. Ahlswede, N. Cai, R. Li, and R. Yeung. Network Information Flow. *IEEE Trans on Information Theory*, 46(4), Jul 2000.

[3] S. Biswas and R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. *ACM SIGCOMM Computer Communication Review*, 34(1):69–74, Jan 2004.

[4] S. Biswas and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *Proc. of ACM SIGCOMM*, Aug 2005.

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *Proc. of ACM SIGCOMM*, Aug 2007.

[6] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proc. of ACM MOBICOM*, Sep 2003.

[7] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *Proc. of ACM MOBICOM*, Sep 2004.

[8] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Least-Cost Opportunistic Routing. In *Proc. of Allerton*, 2007.

[9] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-power Wireless Sensor Networks. Technical Report UCLA/CSD-TR 02-0013, UCLA CS, Dec 2002.

[10] C. Gkantsidis, W. Hu, P. Key, B. Radunovic, P. Rodriguez, and S. Gheorghiu. Multipath Code Casting for Wireless Mesh Networks. In *Proc. of ACM CoNEXT*, Dec 2007.

[11] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *Proc. of IEEE INFOCOM*, 2003.

[12] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *IEEE ISIT*, 2003.

[13] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley-Interscience, 1991.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in The Air: Practical Wireless Network Coding. In *ACM SIGCOMM*, 2006.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Trans. on Computer Systems*, 18(3):263–297, Aug 2000.

[16] D. Koutsonikolas, Y. C. Hu, and C.-C. Wang. XCOR: Synergistic Interflow Network Coding and Opportunistic Routing. In *Poster in ACM MOBICOM*, 2008.

[17] N. Letor, P. De Cleyn, and C. Blondia. Enabling Cross Layer Design: Adding the MadWifi Extensions to Nsclick. In *Proc. First International Workshop on Network Simulation Tools 2007*, Oct 2007.

[18] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched Networks: A New Paradigm for Wireless Transport. In *NSDI*, 2009.

[19] Y. Lin, B. Li, and B. Liang. CodeOR: Opportunistic Routing in Wireless Mesh Networks with Segmented Network Coding. In *IEEE ICNP*, 2008.

[20] MadWifi. User Documentation. http://madwifi.org/wiki/UserDocs.

[21] A. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained Path Selection for Wireless LANs. In *ACM MobiSYS*, Jun 2004.

[22] M. Neufeld, G. Schelle, and D. Grunwald. Nsclick User Manual. Technical report, University of Colorado, Boulder, CO 80309, Aug 2003.

[23] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-Based Models of Delivery and Interference in Static Wireless Networks. In *Proc. of ACM SIGCOMM*, Sep 2006.

[24] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu. Simple Opportunistic Routing Protocol for Wireless Mesh Networks. In *IEEE WiMesh*, 2006.

[25] I. Stojmenovic, A. Nayak, J. Kuruvila, F. Ovalle-Martinez, and E. Villanueva-Pena. Physical Layer Impact on the Design and Performance of Routing and Broadcasting Protocols in Ad Hoc and Sensor Networks. *Computer Communications*, 28(10):1138–1151, Jun 2005.

[26] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[27] A. Warrier, S. Ha, P. Wason, and I. Rhee. DiffQ: Differential Backlog Congestion Control for Multi-hop Wireless Networks. In *Proc. of IEEE INFOCOM*, 2009.

[28] K. Zeng, W. Lou, and H. Zhai. On End-to-End Throughput of Opportunistic Routing in Multirate and Multihop Wireless Networks. In *Proc. of IEEE INFOCOM*, Apr 2008.