SlideOR: Online Opportunistic Network Coding in Wireless Mesh Networks

Yunfeng Lin, Ben Liang, Baochun Li

***Abstract*—Opportunistic routing significantly increases unicast throughput in wireless mesh networks by effectively utilizing the wireless broadcast medium. With network coding, opportunistic routing can be implemented in a simple and practical way without resorting to a complicated scheduling protocol. Traditionally, due to the constraints of computational complexity, a protocol utilizing network coding needs to partition the data into multiple segments and encode only packets in the same segment. However, it is extremely challenging to decide the optimal time to move to the transmissions of the next segment, and existing designs all resort to different heuristic ideas that might harm network throughput. To address this problem, we propose *SlideOR*, a new protocol to encode source packets in overlapping sliding windows such that coded packets from one window position may be useful towards decoding the source packets inside another window position. Through extensive simulations, we show that *SlideOR* outperforms the existing solutions and is amenable to much simpler implementation than solutions with complicated scheduling among multiple segments.**

I. Introduction

One of the fundamental challenges in wireless mesh networks is to maximize the throughput of unicast communication sessions. The traditional wisdom in designing mesh unicast routing protocols (*e.g.*, [1]) treats a wireless link as a point-to-point link, and utilizes a single path to transmit data packets from the source to the destination. It has essentially neglected the fact that a wireless communication channel is a broadcast medium in nature.

In contrast, *opportunistic routing* [2], [3] is able to substantially improve unicast throughput in wireless mesh networks, by effectively utilizing the shared wireless broadcast medium. In opportunistic routing protocols, all neighboring nodes of a transmitter may overhear the data packet and may assist forwarding the packet to its destination. With opportunistic routing, however, a delicate balance of tradeoffs has to be maintained. On the one hand, since multiple nodes in a wireless broadcast region are able to overhear identical copies of a packet, one needs to avoid unnecessary forwarding of duplicates. On the other hand, if too few nodes help to forward the packet, a forwarding path from the source to the destination may not be found. Biswas *et al.* [2], for example, have designed a complex packet scheduling algorithm, with a large number of control messages, in order to achieve such a balanced tradeoff.

The authors are affiliated with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are {*ylin,bli*}*@eecg.toronto.edu* and *liang@comm.utoronto.ca*. This work is supported by NSERC Discovery, CRD and Strategic Grants (RGPIN 262018-07, RGPIN 238994-06, CRDPJ 379623-08, STPGP 364910-08).

In [3], Chachulski *et al.* has shown that with the help of *random network coding*, opportunistic routing can be achieved without complex scheduling, while still taking full advantage of the wireless broadcast medium. The fundamental insight is that, with random mixing of coded packets, although multiple receivers overhear the same packet in a wireless broadcast neighborhood, they are able to generate linearly independent coded packets with high probability, by combining the received coded packet with existing packets in their buffers.

However, due to the constraints of computational complexity involved in random linear codes, it is infeasible to apply network coding to a large number of data packets [3]. The MORE protocol proposed in [3], for example, divides the data stream into different *segments*, and performs coding operations on packets within the same segment. This is usually referred to as *segmented network coding*. Although segmented network coding attempts to achieve a tradeoff between the benefit of network coding and its complexity, it introduces an open challenge that severely affects throughput: *When shall the source stop transmitting coded packets of one segment and move on to the next one?* Intuitively, if the source stops prematurely, the destination may fail to receive a sufficient number of coded packets to decode the entire segment. If the source stops too late, however, a substantial waste of bandwidth resources ensues, since coded packets that are no longer useful to the destination are still being sent by the source.

MORE uses a simple strategy as a stop-gap measure to answer this question: the source simply continues to transmit coded packets belonging to the same segment until an explicit acknowledgment from the destination has been received. Such a "stop-and-wait" protocol reflects one extreme of the tradeoff, where the source stops its transmission *too late*, leading to wasted wireless bandwidth. To partially address this problem, we previously proposed CodeOR [4], which transmits multiple segments in a pipelining fashion, utilizing more timely ACKs from closer nodes in the network to reduce the penalty of inaccurate timing in transmitting the next segment. However, CodeOR uses segmented network coding and suffers from the same fundamental difficulty as MORE.

In this work, inspired by online network coding [5], [6], we propose *SlideOR* to fundamentally address this challenge. In SlideOR, as opposed to segmented network coding, packets are not encoded separately in segments. Instead, the source uses a moving sliding window to determine the set of source packets to be encoded. Hence, in contrast to segmented network coding, where the coded packets in one segment is useless for

the decoding of the next segment, the coded packets belonging to different overlapping sliding windows can be helpful to each other. In this way, the time for window sliding is much less critical and may be optimized for other objectives of the system.

We further discuss two critical challenges in implementing SlideOR. *First*, it is important to decide how far to advance the sliding window when ACKs arrive. If the sliding window moves too slowly such that the source encodes too few new source packet from the new window, there might not be enough original data to send to the next node. On the other hand, if the sliding window moves too fast, there might be too few data packets in common between successive encoding operations, leading to degradation toward segmented network coding. *Second*, a forwarding node may re-encode coded packets from different sliding window locations, such that the number of source packets in the re-encoded packet can be significantly larger than the sliding window size, which leads to increased encoding and decoding complexity. We show that by removing the coded packets with too many outdated source packets, we can maintain low computational complexity while achieving similar or higher throughput than otherwise.

Through extensive simulations, we show that SlideOR achieves substantially higher throughput than existing solutions under the same network setting. In particular, SlideOR achieves two-fold performance gain over MORE [3] and 40% gain over CodeOR [4]. Furthermore, we show that SlideOR is as simple as MORE for implementation, and much simpler than CodeOR since advancing sliding windows is much less challenging than scheduling multiple segments in CodeOR.

The remainder of this paper is organized as follows. We compare SlideOR with related work in Sec. II. In Sec. III, we describe the network model and briefly review network coding with its existing applications in opportunistic routing. In Sec. IV, we describe the protocol design in SlideOR. We use simulations to demonstrate the effectiveness of SlideOR in Sec. V. Finally, Sec. VI concludes the paper.

II. Related Work

ExOR [2] introduces opportunistic routing in wireless mesh networks by utilizing the wireless broadcast medium to increase network throughput, as compared to traditional single-path routing protocols (*e.g.*, [1]), which neglected the wireless broadcast advantage. In order to realize the benefit of opportunistic routing, ExOR introduces a complex packet scheduling algorithm. With random network coding, Chachulski *et al.* [3] have proposed a more practical opportunistic routing protocol, referred to as *MORE*, which is feasible to be implemented in practice, and achieves a higher throughput than ExOR. However, MORE uses a "stop-and-wait" protocol with a single segment in its sending window, which is not efficient utilizing the delay-bandwidth product in large-scale networks.

Similar to our previous work [4], Sundararajan *et al.* [6] combines network coding with TCP Vegas. Our previous work is based on segment based network coding with a predetermined segment size. Such setting complicates protocol design. On the other hand, [6] mainly proposes an end-to-end coding scheme, *i.e.*, the encoding is only operated on the sender, but not on the intermediate nodes in the network. It is easy to show that such an approach significantly degrades the network throughput. Our work extends this idea to combine online network coding with TCP Vegas in multi-hop wireless networks with recoding on intermediate nodes. We also argue the *redundancy parameter* R proposed in [6] may not be necessary, where for every packet arrives from the application level, R linear combinations are sent to the network on average. Instead, in our protocol, the sender keeps injecting more *"redundant"* coded packets into the network until an ACK arrives from the receiver and advances the sliding window.

III. Network Model and Preliminaries

We consider a single unicast communication session in a wireless mesh network, where the source has a stream of data to be transmitted to the destination. We model the wireless network as a directed hypergraph (V, E), where V is the sets of nodes and E is the sets of links. A wireless broadcast link is modeled as a hyper link $(i, J) \in E$, where J is a subset of V. The packet loss events on multiple receivers of a wireless broadcast link are assumed to be independent [3].

Segmented network coding is a special form of random linear network coding. The stream of data to be transmitted from the source is divided into multiple segments, each with a predetermined number of packets. When an arbitrary intermediate node a between the source and the destination wishes to transmit coded packets within a segment, a produces a coded packet x_a by encoding all coded packets in its buffer belonging to the segment, namely $x_1, \ldots, x_m$, where m is the total number of coded packets in the buffer that belong to the segment: $x_a = \sum_{i=1}^{m} \beta_i x_i$, where all multiplication and addition operations are defined on a Galois Field (such as GF(2^8) when the operations are performed on each byte), and β_i is *randomly* chosen from the field. It is easy to see that x_a is also a linear combination of a subset of original packets from the source, and the coefficients can be derived. Node a then broadcasts x_a along with its coding coefficients over the original packets to all its neighbours. The destination collects at least m coded packets for each segment and recovers all m source packets in the segment by solving a set of linear equations. Because the coding coefficients and the coded packet are known, each coded packet represents one linear equation with the m source packets as unknown variables. Gaussian elimination is commonly used to solve this linear system.

In MORE [3], segmented network coding is used with opportunistic routing in the following fashion. The source transmits the segments sequentially and independently, *i.e*, the source only transmits the next segment if the current one is recovered at the destination. Since MORE keeps one segment in the network at any time and wastes network resource when the source is waiting for an ACK for a segment, we propose CodeOR in [4] to address this problem, by transmitting multiple segments simultaneously in the network and by

utilizing the network resource during ACK propagation. To meet the design challenge on deciding how to move from the transmission of one segment to the next, in CodeOR, an intermediate node transmits the next segment when the current segment has been received fully by at least one of its downstream nodes[1]. We will compare MORE, CodeOR, and SlideOR in Sec. IV-B and Sec. V-B.

IV. SLIDEOR: PROTOCOL DESIGN

In this section, we describe the implementation of SlideOR, an opportunistic protocol utilizing online network coding.

A. Baseline Protocol Design

We can decouple SlideOR into three components. First, the encoding and decoding algorithms. Second, the algorithm to advance the sliding window on different nodes. Third, the algorithm to determine when to allow node transmissions. For the third component, we utilize the credit assignment algorithm in [3].

1) Encoding and decoding: Assuming each source packet from the data stream has a sequence number i. At any time, when the MAC layer allows the source to transmit, the source generates a coded packet by randomly encoding a subset of consecutive source packets, referred to as a *sliding window*. We use W to denote the size of the sliding window. Hence, the encoding algorithm to produce a coded packet x at the source is $x = \sum_{j=s}^{s+W} \alpha_j E_j$, where E_j is the jth source packet, α_j is a randomly chosen coefficient from a Galois field such as GF(2^8), and s is the sequence number of the next expected source packet to be received at the destination. We will provide more details about s when we present the decoding algorithm at the destination. The source broadcasts each coded packet x with the associated coding coefficients α_j.

For any forwarding node a between the source and the destination, it accumulates and caches a set of coded packets in its buffer. Upon receiving a coded packet c, node a first checks whether c is linearly dependent with the coded packets in its buffer. If not, node a inserts c into its buffer. Whenever possible, node a generates a new coded packet according to the following algorithm: $x_a = \sum_{i=1}^{m} \beta_i x_i$, where x_i is the ith coded packet in the buffer of node a, and β_i is a coefficient randomly chosen from the Galois field. It is easy to see that x_a is also a linear combination of a subset of original packets from the source, and the coefficients can be derived. Node a then broadcasts x_a along with its coding coefficients over the original packets to all its neighbors.

The destination attempts to decode the incoming coded packets according to the following algorithm. Since each coded packet represents a linear equation with the source packets from which it is encoded as unknown variables, decoding the original source packets is equivalent to solving a linear system composed of all coded packets received so far. The *decoding matrix* represents the coefficient matrix of such a linear system. The destination uses Gauss-Jordan elimination to convert the decoding matrix to a reduced row echelon form and solve the linear system. Fig. 1 shows an example of the processed decoding matrix. In particular, we note that as the decoding proceeds, more and more source packets are decoded such as packets 1, 2, and 3 in the figure. Furthermore, although a fraction of source packets (packet 4 and 5 in the figure) cannot be decoded right away, they can be decoded later as more coded packets encoded with them are coming from the source. We refer to the maximum sequence number of the source packet that can be decoded in the future as the maximum *seen* packet [5], [6]. In other words, the sequence number of the seen packet is essentially the maximum rank of the decoding matrix. We use the notation s to denote the sequence number of the next expected packet at the destination: the sequence number of the packet next to the maximum seen packet. We note that the oldest source packet to be encoded at the source is s as described previously in this section, since it is not necessary to encode any source packets that can already be decoded at the destination.

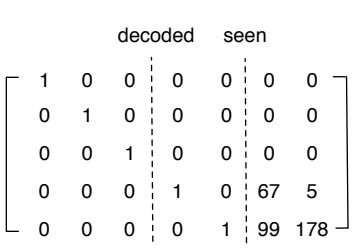


Fig. 1. Packet 1, 2 and 3 are decoded. The destination cannot decode packet 4 and 5 now, but can be *seen* [5].

2) Basic Sliding Window Algorithm: Upon receiving an innovative coded packet, the destination sends an ACK to the source via the shortest path from the destination to the source. The ACK packet carries s, the sequence number of the next expected packet at the destination. s is used to advance the sliding window at the source and indicates the source packet to be encoded with the lowest number at the source. Note that the forwarding nodes do not participate in the sliding window advancement. They only accept coded packets, re-encode, and transmit them.

B. Comparison Among Protocols

We are now ready to compare the basic operation of SlideOR with that of MORE and CodeOR. Fig. 2 illustrates the packet transmission dynamics of these three protocols in a chain network. As shown in Fig. 2(a), for MORE, when the distance between the source and the destination is much longer than the segment size, there might be only a fraction of nodes transmitting the data in one segment. CodeOR, as shown in Fig. 2(b), improves the protocol by allowing more than one segment in transmission. However, it is easy to see that there are challenges to schedule transmissions of multiple segments. In contrast, SlideOR, as shown in Fig. 2(c), encodes and decodes packets in sliding windows such that different windows might overlap. Hence, although SlideOR encodes only a fraction of packets together to achieve low computation

[1]More precisely, the condition is that the current segment has been received *sufficiently* by its downstream nodes. Interested readers are referred to [4] for details.

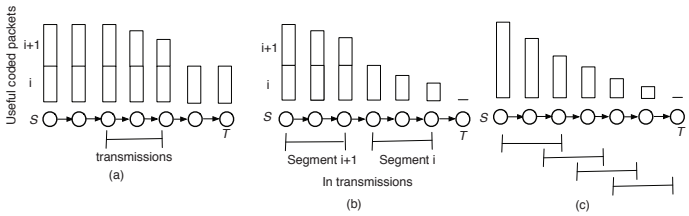


Fig. 2. The height of the bars on each node indicates the number of useful coded packets at each node. Useful transmission (of independent coded packet) is possible only if the height of a sending node is higher than that of a receiving node. (a) MORE (b) CodeOR (c) SlideOR.

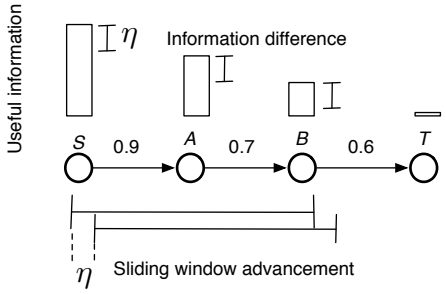


Fig. 3. The advancement η of sliding window should be sufficiently large to create enough information difference for useful packet transmission between adjacent nodes. The number on each link indicates the probability of success at each transmission attempt.

complexity, it might approach the performance of a protocol with an infinite segment size. Furthermore, in SlideOR, there is no complication in scheduling multiple segments, which has been shown to be difficult to implement [4].

C. Advancing Sliding Windows

In this section, we describe the motivation and design of our sliding window advancement algorithm. For simplicity in illustration, we assume that ACK is reliably transmitted from the destination to the source. Since the destination only transmits an ACK when the rank of the decoding matrix is increased by one from receiving an innovative coded packet, when an ACK arrives, the source knows that the sequence number s of the next expected source packet also is increased by one. It would be straight forward for the source to advance the sliding window by one from $[s, s+W-1]$ to $[s+1, s+W]$, where W is the sliding window size. However, such an approach does not work well in reality. We illustrate the reason in Fig. 3.

We first assume that the sliding window is sufficiently small to keep the encoding and decoding complexity reasonably low. Under such a condition, after sending W linearly independent coded packets, the source S starts to transmit linearly dependent coded packets. Note that 0.9 is the successful transmission probability between node S and A. Hence, after an expected number of $W/0.9$ source transmissions, node A receives W coded packets and keeps them in its buffer. From this time on, any coded packets transmitted from the source is redundant to node A. When an ACK from the destination arrives at the source S, if S advances its sliding window by one packet, S then transmits $W+1$ useful source packets, which is only one more than the useful coded packets at node A. Therefore, after one successful packet transmission from S to A, S has no useful data for node A again, and transmits only useless coded packets.

We address this problem by artificially creating *information difference* between adjacent nodes. For instance, when the sliding window advances, the difference between the source S and node A is η, which may be larger than one as shown in Fig. 3. The details of our algorithm is as follows. The source defines l as the lower boundary of its sliding window, *i.e.*, the sliding window is from source packet l to $l+W-1$. When the source receives an ACK indicating that the next expect source packet is packet s. The source advances the sliding window by η packets, such that it spans from the source packets $l+\eta$ to $l+\eta+W-1$, only if $l+\eta >= s$, where η is a positive integer larger than 0 as discussed previously. We refer to η as the *advancement step size* thereafter. It is easy to see that such an algorithm ensures that the information difference between any adjacent nodes is η packets.

We are able to remove obsolete coded packets in buffers, when these packets are no longer useful towards decoding [7]. Similarly, we can remove useless rows and columns in decoding matrix [7]. Furthermore, with the concept of sliding window and ACKs for degree of freedom, it is not hard to utilize TCP Vegas [8] with SlideOR. In particular, SlideOR determines the proper window size similarly to TCP Vegas.

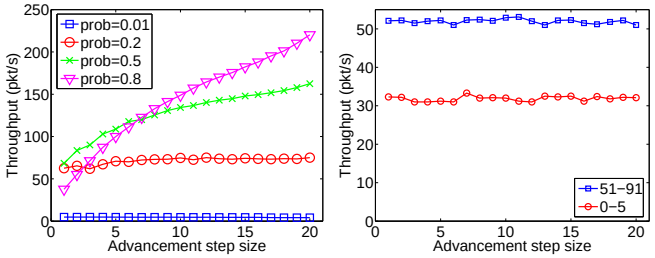
V. Performance Evaluation

We study SlideOR through simulations with a customized discrete event simulator. For the physical layer, we use the measurement-based model from [9] to capture the effect of opportunistic reception in a lossy wireless environment, which empirically maps link distances to transmission success probabilities between two wireless nodes. In our simulations, two nodes are regarded as neighbors only if the link quality between them is sufficient to achieve a transmission success probability higher than 0.05. We conduct experiments on a line topology and a random topology with 100 nodes that are deployed, uniformly at random, in a square of size 4000×4000 square meters.

A. Impact of Advancement Step Sizes

In this section, we investigate the effect of advancement step size η described in Sec IV-C on the protocol performance under different topologies. We set the sliding window size to 20, while varying η. We conduct experiments on the simple line topology with 4 nodes shown in Fig. 3. In addition, in order to study the relation between η and link transmission probabilities, we set the transmission probabilities on all links to 0.01, 0.2, 0.5, and 0.8 for different experiments.

From Fig. 4(a), we observe that the protocol has a larger throughput if we set a larger η. Furthermore, we note that the advantage of a larger η degrades when link transmission probabilities decrease from 0.8 to 0.01. This is because with a higher loss rate, a node needs a longer time to deliver a coded



(a) (b)

Fig. 4. Network throughputs under different advancement step sizes η on (a) the line topology in Fig. 3 with link transmission probabilities 0.01, 0.2, 0.5, and 0.8 and (b) random topology with the source and the destination set to nodes 51 and 91 or nodes 0 and 5.

packet to its neighbor. Hence, a smaller difference between the number of coded packets on adjacent nodes is sufficient to provide useful coded packet transmissions.

In the second set of experiments, we choose two pairs of random nodes on the random topology, and conduct the same experiments. Fig. 4(b) shows that the loss probabilities on the random topology used in this simulation is sufficiently high such that a small η is enough for a high network throughput.

B. Comparison Among Different Protocols

In this section, we compare the performance of MORE, CodeOR, and SlideOR under different network settings. Similarly to previous sections, we use node 51 and 91 as the source and the destination, respectively. Fig. 5 shows the network throughput of CodeOR under segment sizes 2, 5, 10, 25, and 50, with window sizes (the number of segments in a sending window) 25, 10, 5, 2, and 1, respectively. Hence, the total number of coded packets in one window is always 50. We note that when the window size is 1, CodeOR is MORE, since only one segment is in transmission in the network at any time. Fig. 5 shows the throughput of CodeOR. We observe that when the segment size is very small, *e.g.*, 2, its throughput suffers. This is because network coding is not effective with a smaller segment size, which, in essence, means less packet mixing and a larger amount of segment scheduling overhead. On the other hand, when the segment size increases from 10 to 50, the number of segments decreases and the throughput degrades since CodeOR is less flexible with transmissions during ACK propagation. In the extreme case, the protocol degrades to MORE when the segment size is 50, and there are no useful transmissions at all when ACKs are propagated as we discussed in [4].

We then conduct experiments for SlideOR with sliding window sizes 2, 5, 10, 25, and 50, corresponding to the experiment settings for CodeOR. The advancement step size η is set as roughly half of the sliding window size: 1, 2, 5, 12, and 25, respectively. We observe from Fig. 5 that the throughput of SlideOR increases when its sliding window size increases. In particular, when the sliding window increases to a size larger than 25, SlideOR outperforms CodeOR. We emphasize that the total number of coded packets sent by MORE and CodeOR is always 50, whereas the sliding window size is under 50 most of the time for SlideOR. Hence, all these

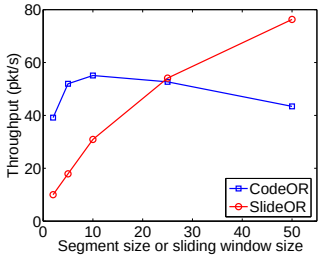


Fig. 5. Network throughputs under different segment/sliding-window sizes.

comparisons are unfair to SlideOR except for the last data point. However, when the sliding window size is 50, SlideOR significantly outperforms MORE, since it can transmit useful coded packets simultaneously during ACK propagation, whereas MORE cannot. SlideOR achieves almost the same performance as CodeOR even when its sliding window, 25, is half of the total number of packets in the sending window, 50, of CodeOR because it eliminates imperfect node scheduling between segments in CodeOR [4].

VI. Conclusion

In this paper, we propose SlideOR, an opportunistic routing protocol utilizing online network coding. We describe the challenges and solutions in implementing such a protocol. In addition, we show that SlideOR significantly outperforms existing opportunistic protocols based on network coding, through extensive experiments. SlideOR can achieve much higher throughput, because it transmits useful coded packets during ACK propagation, while preventing the scheduling overhead as in some of the existing solutions. At the same time, SlideOR is substantially simpler and much easier to implement, by reducing the complication of scheduling multiple segments to simple sliding window advancements.

References

[1] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network," in *Proc. of ACM MOBICOM*, 2005.

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

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

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

[5] J. K. Sundararajan, D. Shah, and M. Medard, "ARQ for Network Coding," in *Proc. of IEEE International Symposium on Information Theory*, 2008.

[6] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network Coding Meets TCP," in *Proc. of IEEE INFOCOM*, 2009.

[7] Y. Lin, B. Liang, and B. Li, "SlideOR: Online Opportunistic Network Coding in Wireless Mesh Networks," http://iqua.ece.toronto.edu/papers/slideor.pdf, ECE, University of Toronto, Tech. Rep., December 2009.

[8] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Area in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[9] J. Camp, J. Robinson, C. Steger, and E. Knightly, "Measurement Driven Deployment of a Two-Tier Urban Mesh Access Network," in *Proc. of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2006.