Pacifier: High-Throughput, Reliable Multicast without "Crying Babies" in Wireless Mesh Networks

Dimitrios Koutsonikolas, Y. Charlie Hu, Chih-Chun Wang
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907
{dkoutson, ychu, chihw}@purdue.edu

***Abstract*—In contrast to unicast routing, high-throughput reliable multicast routing in wireless mesh networks (WMNs) has received little attention. There are two primary challenges to supporting high-throughput, reliable multicast in WMNs. The first is no different from unicast: wireless links are inherently lossy due to varying channel conditions and interference. The second, known as the "crying baby" problem, is unique to multicast: the multicast source may have varying throughput to different multicast receivers, and hence trying to satisfy the reliability requirement for poorly connected receivers can potentially result in performance degradation for the rest of the receivers.**

In this paper, we propose *Pacifier*, a new high-throughput reliable multicast protocol for WMNs. *Pacifier* seamlessly integrates four building blocks, namely, *tree-based opportunistic routing, intra-flow network coding, source rate limiting, and round-robin batching,* to support high-throughput, reliable multicast routing in WMNs, while at the same time effectively addresses the "crying baby" problem. Our evaluations show that *Pacifier* increases the average throughput over a practical, state-of-the-art reliable network coding-based protocol MORE by 171%, while improving the throughput of well-connected receivers by up to a factor of 20.

I. Introduction

Wireless mesh networks (WMNs) are increasingly being deployed for providing cheap, low maintenance Internet access (e.g. [1], [2], [3]). These networks have statically deployed mesh routers that are not energy constrained, and hence the main design challenge is to improve applications' performance, in particular, to provide high throughput and reliability in network access. Indeed, recent years have witnessed numerous "exotic" protocols that aim to improve the throughput and reliability of unicast routing. These include opportunistic routing (OR) protocols (e.g., [4]), protocols that exploit inter-flow (e.g., [5]) or intra-flow (e.g., [6]) network coding, as well as lower layer protocols (e.g., [7]).

In contrast to unicast routing, high-throughput, reliable multicast routing has received relatively little attention. Reliable multicast routing has many important applications in WMNs, such as software updates and video/audio file downloads. These applications have a strict requirement of **100% Packet Delivery Ratio (PDR)**, since every byte of the downloaded file has to be received by **all** the receivers. This requirement makes many of the reliable multicast protocols proposed in the past (e.g., [8], [9], [25]) inappropriate, since they cannot guarantee 100% PDR. In addition, reliability for this class of applications cannot come at the cost of significantly reduced throughput, unlike in military applications [8], since Internet users always desire fast downloads.

The fundamental challenge in achieving reliable multicast in WMNs is no different from that of reliable unicast – that wireless links are lossy. To overcome this, researchers have applied classic techniques such as Automatic Repeat reQuest (ARQ), Forward Error Correction (FEC), or combinations of the two. The majority of the works on reliable multicast in multihop wireless networks either are solely based on ARQ (e.g., [10], [11]) which suffer the feedback implosion problem, or combine ARQ with congestion control (e.g., [8], [12], [13]). A recent work [14] studied the applicability of FEC and hybrid ARQ-FEC techniques, borrowed from the wired Internet, to WMNs, and showed that RMDP [15], a hybrid ARQ-FEC protocol, can achieve both reliability and high throughput.

More recently, researchers have applied network coding (NC), a technique originally developed for the wireline Internet, to overcome the above challenge. [16] showed that the operation of mixing packets resembles the operation of rateless FEC codes. Actually, NC can be viewed as a technique equivalent to performing hop-by-hop FEC, without the delay penalty incurred by the decoding operations at each hop, that would be required by hop-by-hop FEC. In [17], the authors went one step further and showed that the reliability gain (expressed as the expected number of transmissions) of NC over end-to-end FEC for a wireless multicast tree of height h with link loss rate p is in the order of $\Theta((\frac{1}{1-p})^h)$.

Practical work that exploits the idea of utilizing NC for reliable multicast is still at a preliminary stage. MORE [6] is the *only practical* NC-based protocol that supports high-throughput, reliable multicast. It combines NC with opportunistic routing, with the primary goal of removing the need for coordination required in opportunistic routing. However, the design of MORE also guarantees reliability, i.e., MORE is a routing protocol for *reliable file transfer*, for both unicast and multicast.

A second fundamental challenge in reliable multicast, which is unique to multicast, is the "crying baby" problem as first pointed out in [18] in the context of multicast in the Internet. If one receiver has a particularly poor connection, then trying to satisfy the reliability requirement for that receiver may result in performance degradation for the rest of the receivers. This problem also raises the interesting question of what is a suitable definition of overall performance metric if multiple receivers are allowed to achieve uneven throughput.

Regardless, a major challenge in the design of high throughput, reliable multicast protocols is whether it is possible to develop a protocol that improves the throughput of well-connected receivers without worsening the already low throughput of poorly-connected receivers.

In this paper, we propose *Pacifier*, a high-throughput, reliable multicast protocol that systematically addresses the above two challenges. *Pacifier* seamlessly integrates four building blocks, namely, *tree-based opportunistic routing, intra-flow NC, source rate limiting, and round-robin batching*, to support high-throughput, reliable multicast routing and at the same time solve the "crying baby" problem. First, *Pacifier* builds an efficient multicast tree traditionally used by multicast protocols and naturally leverages it for opportunistic overhearing. Second, *Pacifier* applies intra-flow, random linear NC to overcome packet loss over lossy links which avoids hop-by-hop feedback and the coordination of multicast tree forwarders in packet forwarding. Third, *Pacifier* applies rate limiting at the source, reducing the congestion level in the network. Fourth, *Pacifier* solves the "crying baby" problem by having the source send batches of packets in a round-robin fashion. This functionality allows *Pacifier* to improve the throughput of well-connected nodes drastically and often times of poorly-connected nodes. The reason for the later is that as more and more receivers complete decoding, the source can prune the tree branches towards them, reducing the number of FNs and the amount of contention in the network.

We evaluate *Pacifier* and compare its performance against MORE, using extensive realistic simulations. Our simulation results show that *Pacifier* increases the average throughput of multicast receivers over MORE by 171%, while it solves the "crying baby" problem, by increasing the maximum throughput gain for well-connected receivers by up to 20x. Interestingly and importantly, *Pacifier* also improves the throughput of the "crying babies", i.e., the poorly connected receivers, by up to 4.5x.

To our best knowledge, *Pacifier* is the first practical multicast protocol that simultaneously satisfies both requirements posed by protocols designed for commercial WMNs: it guarantees 100% PDR, while simultaneously offering significant throughput improvements for *all* receivers over state-of-the-art protocols. While the design of *Pacifier* is based on the numerous principles developed over the past fifteen years in the field of reliable multicast, the use of NC makes the integration of these techniques much simpler and more efficient. Finally, *Pacifier* uses the same type of NC as MORE, and has the same memory requirements at the routers, and hence, like MORE, it can be easily implemented on commodity hardware.

II. Related work

In spite of the extensive research on reliable multicast in the wired Internet, which went through the development of ARQ-based schemes (e.g., [19], [18]), to FEC schemes (e.g., [20]), to hybrid ARQ-FEC schemes (e.g., [21], [15], [22]), to rateless codes (e.g., [23]), the majority of the work on reliable multicast in multihop wireless networks have used the traditional ARQ techniques. A survey on reliable multicast protocols for ad hoc networks [24] classifies them into deterministic and probabilistic ones, depending on whether data delivery is fully reliable or not. Deterministic protocols (e.g., [10], [8], [12], [11]) provide deterministic guarantees for packet delivery ratio, but they can incur excessive high overhead and drastically reduced throughput. On the other hand, probabilistic protocols (e.g., [9], [25]) incur much less overhead compared to the former, but they do not offer hard delivery guarantees. Using rateless codes requires the source to continuously send packets, which can cause congestion in the bandwidth-limited wireless networks. Recently, [14] studied the applicability of FEC and hybrid ARQ-FEC techniques, borrowed from the wired Internet, to WMNs, and showed that RMDP [15], a hybrid ARQ-FEC protocol, can provide both reliability and high throughput.

Most recently, intra-flow network coding (NC) has been proposed as a whole new approach to reliable routing. NC in theory is equivalent to hop-by-hop FEC [16], [17], and hence the maximum amount of redundancy injected from any node in the network is determined by the lossiest link of the tree, and not by the lossiest path from the source to any receiver, unlike in end-to-end FEC. However, hop-by-hop FEC/NC also has its practical drawbacks; it requires buffering packets at each node for decoding/re-encoding (in case of FEC) or only re-encoding (in case of NC). Due to the constraints on the buffer size and on packet delay, NC needs to send packets in batches, i.e., the source needs to wait till a batch is received by all receivers before proceeding to the next batch. This introduces the "crying baby" problem, where the poorly connected receivers slow down the completion time of well-connected receivers.

To our best knowledge, MORE is the only NC-based protocol for high-throughput, reliable multicast (though it is also for unicast). Due to its significance, and since we will compare *Pacifier* against it in our evaluation, we present a brief overview of MORE below. To our knowledge, the only other practical NC-based multicast protocol is CodeCast [26], which exploits NC for *improving* but not *guaranteeing* reliability in multimedia multicast applications in mobile ad hoc networks.

A. Overview of MORE

MORE [6] is an opportunistic routing protocol for reliable file transfer. MORE is implemented as a shim between the IP and the 802.11 MAC layer. We briefly review its two major features: forwarding node (FN) selection and packet batching.

FN selection. MORE uses the ETX metric [27], based on loss rate measurements, to select the possible FNs. For each destination the source includes in the FN list the nodes whose ETX distance to that destination is shorter than the source's distance. Also, for each FN the source includes a *TX_credit* in the FN list. The *TX_credit* is the expected number of transmissions a node should make for every packet it receives from a node farther from a destination in the ETX metric, in order to ensure that at least one node closer to the destination will receive the packet.

The algorithm for FN selection and TX_credit calculation is run at the source. The algorithm starts by assuming that *every* node is a candidate FN for a source-destination pair and calculates the expected number of transmissions this node would make. It then prunes nodes that are expected to perform less than 10% of the total transmissions and assigns TX_credits to the remaining ones, which form a belt of FNs that connect the source to the destination. The algorithm is repeated for each destination; in the end the belts formed for each destination are merged into the final FN set. If an FN belongs to more than one belts, the algorithm calculates a different expected number of transmissions for each of the belts it belongs to. Its final TX_credit is then calculated using the maximum number of transmissions among these belts.

Batching and Coded Packet Forwarding. In MORE, the source breaks a file into batches of k packets. Whenever the MAC is ready to send a packet, the source creates a random linear combination of the k packets of the current batch and broadcasts the encoded packet. Each packet is augmented with its code vector, the batch ID, the source and destination IP addresses and the list of FNs for that multicast, with their TX_credits.

Packets are broadcast at the MAC layer, and hence they can be received by all nodes in the neighborhood. When a node hears a packet, it checks if it is in the packet's FN list. If so, the node checks if the packet is *linearly independent* with all the packets belonging to the same batch that it has already received. Such packets are called *innovative packets* and are stored in a buffer. Non-innovative packets are discarded. Every time a node receives a packet from an upstream node, it increments its *credit_counter* by its assigned TX_credit included in the packet header. If its credit_counter is positive, whenever the MAC is ready to send a packet, the node creates a linear combination of the innovative packets it has received so far[1] and broadcasts it. Broadcasting a packet decrements the credit_counter by one unit.

Finally, a multicast receiver decodes a batch once it collects k innovative packets from that batch. It then sends an ACK back to the source along the shortest ETX path in a reliable manner.

III. *Pacifier* Design

The design of *Pacifier* addresses several weaknesses of MORE. In particular, the belt-based forwarding in MORE can be inefficient for multiple receivers, MORE lacks source rate limiting which can lead to congestion in data dissemination, and MORE suffers the "crying baby" problem.

For clarity, we present the design of *Pacifier* in several steps. We first present a basic version of *Pacifier*, which consists of several building blocks: tree-based opportunistic multicast routing, batching and network coding-based forwarding, credit calculation. The basic version guarantees reliability and already increases throughput compared to MORE. We then present two more optimizations: source rate limiting which avoids congestion and further improves the throughput, and round-robin batching, which solves the "crying baby" problem.

A. Tree-based Opportunistic Routing

We argue that the use of opportunistic routing in the form used in MORE is an overkill for multicast and it can lead to congestion, for two reasons. First, even for a single destination, congestion can occur if too many nodes act as FNs, or if the FNs are far from each other and they cannot overhear each other's transmissions [28]. The situation is worsened when the number of flows increases, since almost all nodes in the network may end up acting as FNs. Such performance degradation was observed in the evaluation of MORE in [6] for many unicast flows; the situation is not very different for many hypothetical unicast flows from a source to many multicast receivers. Second, the benefit of overhearing of broadcast transmissions, which is explored by opportunistic routing in MORE, is naturally explored in a fixed multicast tree, where the use of broadcast allows nodes to receive packets not only from their parent in the multicast tree, but also from ancestors or siblings, essentially transforming the tree into a mesh. We note this property of opportunistic reception of broadcast transmissions has been previously exploited in the design of some of the first multicast protocols for multihop wireless networks (e.g., ODMRP [29]), for improving the PDR.

The above observation motivates a simple multicast-tree based opportunistic routing design. Specifically, *Pacifier* starts by building a multicast tree to connect the source to all multicast receivers. The tree is a shortest-ETX tree, constructed at the source by taking the union of all the shortest-ETX paths from the source to the receivers, which in turn are based on periodic loss rate measurements.[2] The multicast tree is reconstructed at the source every time some receiver completes a batch (Section III-A1) and notifies the source.

1) Batching and Coded Forwarding: As in MORE, the source and the intermediate FNs in *Pacifier* use *intra-flow* random linear NC. The hop-by-hop nature of NC requires the source to break a file into small batches of packets so that the packet header overhead, encoding/decoding time, and memory requirements at the intermediate FNs remain low. We selected a batch size of $k = 32$ packets in *Pacifier*, same as in [6], [30]. For each batch, the source sends random linear combinations of the packets belonging to that batch. The random coefficients for each linear combination are selected from a Galois Field of size 2^8, again same as in [6]. Intermediate FNs store all the innovative packets of the batch and also send random linear combinations of them. Every transmitted encoded packet is augmented with its coding vector, i.e., the vector of the random coefficients used to generate that packet. When a receiver receives any k linearly independent coded packets of a batch, it decodes the batch to obtain the k original packets. It then

[1] Linear combinations of encoded packets are also linear combinations of the original packets.

[2] As [6] argues, periodic link loss rate measurements and their distribution to all nodes in the network is required in all state-of-the-art routing protocols, and the overhead this process incurs is not considered *Pacifier*-specific.

sends an ACK back to the source along the shortest ETX path in a reliable manner.

To achieve reliability, this basic version of *Pacifier* uses the following *batch termination scheme*: the source keeps transmitting packets from the same batch, until *all* the receivers acknowledge decoding of this batch. Such a transmission scheme however introduces the "crying baby" problem as the completion time of each batch is limited by that of the worst receiver.

2) How many packets does an FN send?: Despite the use of a multicast tree for data forwarding, the use of 802.11 broadcast effectively enables opportunistic routing, i.e., a node can opportunistically receive packets from nodes other than its parent in the multicast tree. If a node forwards every packet it receives, a receiver could potentially receive each packet originated from the source multiple times. To avoid unnecessary transmissions, we need to carefully analyze *how many (coded) packets an FN should send upon receiving a data packet.*

Since, in practice, an FN should be triggered to transmit only when it receives a packet, we derive the number of transmissions each FN needs to make for every packet it receives. We define this number as the TX_credit for that FN. Thus, in *Pacifier*, an FN node j keeps a credit counter. When it receives a packet from an *upstream* node (defined below), it increments the counter by its TX_credit. When the 802.11 MAC allows the node to transmit, the node checks whether the counter is positive. If yes, the node creates a coded packet, broadcasts it, then decrements the counter. If the counter is negative, the node does not transmit. We note that opportunistic reception of data packets is always allowed, even from downstream nodes. The credit calculation is on how many packets to be transmitted by the FN upon receiving a data packet from an upstream node.

In the analysis, we focus on disseminating one data packet from the root down the multicast tree. Our analysis is based on the simple principle that in disseminating a packet from the root, each FN in the multicast tree should ensure that each of its child nodes receives the packet *at least once*. Note this principle slows down a parent node to wait for the worst child and creates the "crying baby" problem at each FN, but is consistent with the batch termination scheme of this basic version of *Pacifier*.

We assume an FN j sends packets after receiving from any nodes with lower ETX distance from the root to them, i.e., j's upstream nodes. These nodes are likely to receive packets from the root before j.[3] We also assume that wireless receptions at different nodes are independent, an assumption that is supported by prior measurements [31].

Let N be the number of FNs in the multicast tree rooted at s. Let ϵ_{ij} denote the loss probability in sending a packet from node i to node j. Let z_j denote the expected number of transmissions that FN j must make in disseminating one packet (from the root) down the multicast tree. Let $C(j)$ denote the set of child nodes of j in the multicast tree, and $A(j)$ denote the set of j's upstream nodes.

The expected number of packets that j receives from ancestor nodes is $\sum_{i \in A(j)} z_i(1-\epsilon_{ij})$. Recall j's objective is to make sure each of its child nodes receives at least *one* packet. Since each child node $k \in C(j)$ has already overheard $\sum_{i \in A(j)} z_i(1-\epsilon_{ik})$ from node j's ancestors, the amount of packets node j actually needs to forward for child k is:

$$L_{jk} = min(\sum_{i \in A(j)} z_i(1-\epsilon_{ij}), 1) - \sum_{i \in A(j)} z_i(1-\epsilon_{ik}) \qquad (1)$$

The min operation ensures that j does not forward the same packet more than once, in case it receives it from more than one FNs. Note for the source node s, $L_{sk} = 1$ for all $k \in C(s)$.

Since the expected number of times node j has to transmit a packet to ensure that its child k will receive one packet is $\frac{1}{1-\epsilon_{jk}}$, the expected number of transmissions of j for child k to receive L_{jk} is:

$$z_{jk} = \frac{L_{jk}}{1-\epsilon_{jk}} \qquad (2)$$

Since packets are broadcast, they can be received by more than one child nodes at a time. Hence, the expected number of transmissions node j has to make to ensure that each child node has *one* packet is:

$$z_j = max_{k \in C(j)} z_{jk} \qquad (3)$$

z_j and L_{jk} are inter-dependent, and can be calculated recursively in $O(N^2)$ operations, i.e., by traversing the FNs in the increasing order of their ETX values from the source. Since the order of FNs is well-defined, there are no loops in the credit calculation.

For each data packet the source sends down the multicast tree (which may require multiple transmissions), FN j receives $\sum_{i \in A(j)} z_i(1-\epsilon_{ij})$. Thus, the TX_credit of node j is:

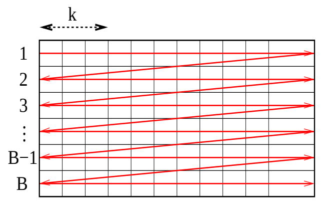
$$\text{TX\_credit}_j = \frac{z_j}{\sum_{i \in A(j)} z_i(1-\epsilon_{ij})} \qquad (4)$$

B. Source Rate Limiting

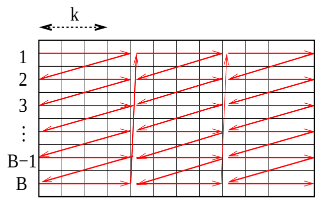
Recent studies have shown the importance of adding rate control to NC-based unicast routing protocols, which exploit MAC layer broadcast [30], [32]. However, end-to-end rate control in multicast is much more complex than in unicast, and there is no widely accepted solution so far. In *Pacifier*, the use of TX_credits implements a form of rate control at which each intermediate FN injects packets into the network. However, the source can potentially send out all the packets in a batch unpaced.

To add rate control to the source, we exploit the broadcast nature of the wireless medium and apply a simple form of backpressure-based rate limiting, inspired by BMCC [13]. The basic idea is to have the source wait until it overhears its child nodes forward the previous packet it sent before it transmits the next packet. Since the number of transmissions by the source

[3] In contrast, MORE's credit calculation was based on the ordering of FNs according to their ETX distance to the destination node. It is unclear that nodes with larger ETX distance to the destination will receive the packet from the root sooner.



(a) Sequential batch transmission in MORE. Each batch is acknowledged by *all* the receivers before the source moves to the next one.



(b) Round-robin batch transmission in *Pacifier*. The source moves to the next batch every time *one* receiver acknowledges the current batch.

Fig. 1. 2 different ways of transmitting B batches of k original packets each: sequential (as in MORE), and round-robin (as in *Pacifier*). For better visualization, we assume here (but not in our actual implementation) that the same total amount of redundancy is required to be sent for each batch.

z_s has already factored in packet losses to its child nodes, the source does not need to worry about losses of individual transmissions, i.e., it does not need to wait until all its child nodes forward each packet it sends out. In fact, it is not even sure that every of its transmissions will trigger a transmission at each of its child nodes, as some nodes may have negative credit counters. Instead, the source waits until it overhears a transmission from *any* of its child nodes or until a timeout.

In [33], the authors suggested a heuristic timeout of $3 \times T_p$ for the backpressure-based unicast protocol, where T_p is the transmission time of one data packet, which depends on the packet size and the MAC data rate. The factor of 3 is to account for the contention time preceding each transmission. Following the same reasoning, in *Pacifier*, we set the timeout to $\sum_{j \in C(s)}$ TX_credit$_j \times 8 \times T_p$. This choice for the timeout reflects the fact that in *Pacifier* a transmission from the source will trigger on average $\sum_{j \in C(s)}$ TX_credit$_j$ transmissions from its child nodes, which in the worst case can be serial, and also the fact that in multicast contention near the source is in general higher.

C. Solving the "Crying Baby" Problem

In MORE, the source keeps transmitting packets from the same batch until all the receivers acknowledge that batch, as shown in Figure 1(a). This policy makes the protocol susceptible to the "crying baby" problem, since if the connection to one receiver is poor, it can slow down the rest of the receivers. The basic version of *Pacifier* we have described so far suffers from the same problem.[4] Note the problem would not exist if the intermediate routers had unlimited memory and hence the whole file were coded into one batch, and there were no constraints on the delay. In the following, we describe a practical solution to the problem, which requires no more memory than MORE or our basic version, i.e., FNs still maintain only one batch at a time in their memory.

[4]BMCC drops the packets on the path towards the worse receiver, in order to prevent that receiver from holding back the rest of the receivers. However, this solution is unacceptable in *Pacifier*, which is designed for applications that require 100% PDR.

In the proposed scheme, the source in *Pacifier* iteratively sends the batches of a file in a *round-robin* fashion, for as many rounds as required, until it has received acknowledgments of receiving all batches from all the receivers, as shown in Figure 1(b). In detail, the source maintains a counter C_{s_i} for each batch i which is equal to the number of remaining packets the source has to transmit for that batch. The counter for batch i is initialized as $C_{s_i} = z_s \times k$, where k is the batch size, and it is decremented every time a packet from batch i is transmitted. Each intermediate FN forwards according to its TX_credit, and only buffers packets belonging to the current batch; when it receives the first packet from a new batch, it flushes its buffer and starts buffering packets from the new batch. The source determines when to switch to work on the next batch as follows. The source sends packets from batch i until either C_{s_i} reaches 0 or it receives from *one* receiver acknowledging completion of this batch;[5] it then moves to the next batch for which there are still receivers that have not acknowledged it. When it finishes with the last batch B, the source starts the next round by going back to the first batch for which it has not received acknowledgments from all receivers. For each such batch it revisits, it recalculates the multicast tree (forwarding nodes) and the TX_credit values for the FNs based on the receivers that have not sent acknowledgments and resets $C_{s_i} = z_s \times k$ using the newly calculated z_s. Effectively, this round-robin batching scheme allows receivers with good connections to the source to quickly obtain the necessary number of packets to decode each batch and complete the file downloading, without waiting for the rest of the receivers.

We note the above round-robin batching scheme is similar to the data carousel first introduced in [22] for an FEC-based protocol. However, the use of NC in *Pacifier* makes this operation much more efficient, since every packet sent is a new linear combination, i.e., there are no duplicates.

1) Adjusting TX_credit Calculation: In the basic version of *Pacifier* (Section III-A2), we defined the TX_credit of an FN as the expected number of packets it has to transmit for every packet it receives from its upstream nodes, in order to ensure that *all* of its child nodes will receive one packet. This definition is consistent with the batch termination scheme of the basic scheme, i.e., the source completes a batch when it receives acknowledgments from all receivers. However, it is inconsistent with the round-robin batching scheme, which aims to prevent poorly-connected receivers from slowing down well-connected receivers. Hence under the round-robin batching, we adjust the definition of TX_credit of an FN to be the expected number of packets it has to transmit for every packet it receives from its upstream nodes, in order to ensure that *at least one* of its child nodes will receive one packet. To realize this change, we simply change the max operator to min in Equation (3). We note this new definition is also consistent with the policy of moving to the next batch whenever any receiver acknowledges the current batch.

[5]Allowing the source to move to the next batch only when it receives an ACK from one receiver is not always efficient, as under heavy congestion, ACKs may delay to reach the source.

2) Intricacies in TX_credit Calculation: There is a subtlety in the above adjustment to the TX_credit calculation under the round-robin batching scheme, i.e., changing the max operator to min in Equation (3). The derivation of Equation (3) is based on the expected number of opportunistic packet receptions (based on the ETX measurements). However, in the actual dissemination of any given batch i, it is possible that the actual packet reception is below or above the expected value. In the later case, the best receiver will successfully receive all packets for that batch, and it is the correct thing to do for the source to move on to the next batch. However, in the former case, the best receiver could be a few packets short of receiving the whole batch i, and hence if the source moves on to the next batch, even the best receiver has to wait for a whole round before the source transmits again packets from batch i. On the other hand, if we had let the source send some additional packets to those predicted by Equation (3), there is a good chance that the best receiver would have also finished in the current round; this would increase the throughput of the best receiver. The challenge here is that it is unknown beforehand whether the opportunistic reception in any particular batch is above or below the expectation, and hence those extra packets sent by the source for a batch can potentially elongate each batch and reduce the throughput of the best receiver.

To facilitate studying the above subtlety in the TX_credit calculation under the round-robin batching scheme, we introduce a tunable knob in Equation (3). Essentially, we define the expected number of transmissions node j makes to its child nodes as $z_j = min_{k \in C(j)} z_{jk} + knob * (max_{k \in C(j)} z_{jk} - min_{k \in C(j)} z_{jk})$. Setting $knob$ to 1 changes the objective to ensuring all child nodes receive a packet at least once, while setting $knob$ to 0 changes the objective to ensuring at least one child node receives a packet at least once. In Section IV-B5, we evaluate the impact of this knob by comparing the performance of *Pacifier* for different values of $knob$.

IV. Evaluation

A. Evaluation Methodology

We evaluated the performance of *Pacifier* and compared it against MORE using extensive simulations. The use of a simulator allowed us to evaluate the performance of the two protocols in large networks, using a diverse set of topologies, which are difficult to create in a testbed. We note *Pacifier* uses the same type of NC and has the same memory requirements and the same fields in the packet header as MORE,[6] and hence it can be easily implemented in practice.

Simulation Setup. We used the Glomosim simulator [34], a widely used wireless network simulator with a detailed and accurate physical signal propagation model. Glomosim simulations take into account the packet header overhead introduced by each layer of the networking stack, and also the additional overhead introduced by MORE or *Pacifier*. For the implementation of MORE, we followed the details in [6].

[6]*Pacifier* only includes the list of FN nodes in the header, sorted in increasing ETX distance from the source. It does not require information about the edges of the tree.

TABLE I
VERSIONS OF MORE AND *Pacifier* EVALUATED IN OUR STUDY. ALL VERSIONS INCLUDE INTRA-FLOW NC.

Name	Description
MORE	MORE [6] optimized with scenario-specific pruning threshold
TREE	Tree-based OR
TREE+RL	Tree-based OR, source rate limiting
TREE+RL+RRB (*Pacifier*)	Tree-based OR, source rate limiting, and round-robin batching

We simulated a network of 50 static nodes placed randomly in a $1000m \times 1000m$ area. The average radio propagation range was 250m, the average sensing range was 460m, and the channel capacity was 2Mbps. The *TwoRay* propagation model was used. To make the simulations realistic, we added fading in our experiments. The Rayleigh model was used, as it is appropriate for WMN environments with many large reflectors, e.g., walls, trees, and buildings, where the sender and the receiver are not in Line-of-Sight of each other. Because of fading, transmission and sensing range are not fixed; they actually vary significantly around their average values.

We simulated each protocol on 10 different randomly generated topologies (scenarios), i.e., placement of the 50 nodes. For each scenario, we randomly generated a multicast group consisting of 1 source and 9 receivers. The source sent a 12MB file, consisting of 1500-byte packets. We present the results for each scenario and the average over all 10 scenarios.

Evaluation Metrics. We used the following metrics:

Average Throughput: The file size (in bytes) divided by the total time required for a receiver to collect the necessary number of packets for decoding, averaged over all receivers.

Total number of data packet transmissions:[7] The total number of data packets broadcast by the source and the FNs.

Source Redundancy: The total number of encoded data packets sent by the source divided by the file size. It gives an estimate of the redundancy injected in the network by the source.

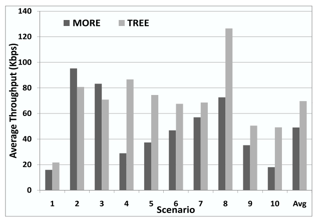
Note that we did not use the PDR as a metric, since all the protocols *guarantee* 100% PDR.

B. Simulation Results

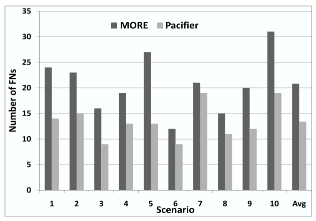
We start by optimizing MORE's pruning strategy as the default strategy appears to cause frequent network partition. We then proceed to evaluate the incremental performance benefit of *Pacifier*'s major components, i.e., the basic version, adding source rate limiting, and adding round-robin batching. Table I summarizes the different versions of MORE and *Pacifier* evaluated.

1) Fixing MORE's pruning threshold: Recall from Section II-A that MORE prunes FNs that are expected to perform less than 10% of the total number of transmissions. We found using such a pruning threshold results in some receivers getting

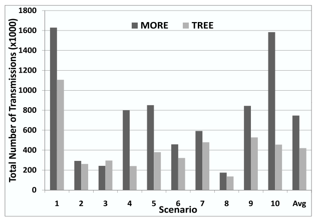
[7]The number of control packets (ACKs) is the same for both MORE and *Pacifier*.



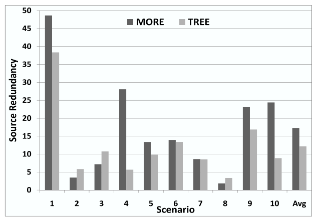
(a) Throughput



(b) Number of FNs

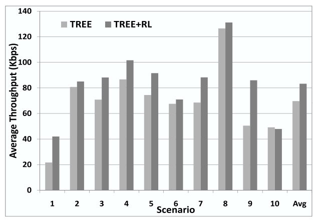


(c) Total # of Transmissions

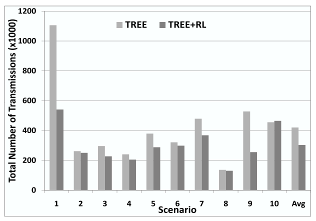


(d) Source Redundancy

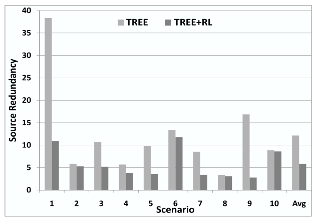
Fig. 2. Throughput, number of FNs, total number of transmissions, and source redundancy with MORE and TREE for 10 different scenarios.



(a) Throughput



(b) Total # of Transmissions



(c) Source Redundancy

Fig. 3. Throughput, total number of transmissions, and source redundancy with MORE and TREE+RL for 10 different scenarios.

disconnected from the source in 8 out of 10 scenarios. Recall also that in MORE, the source proceeds to the next batch only when all receivers acknowledge decoding of the current batch. When a receiver is disconnected, the source will never leave the first batch, and all the receivers will receive zero throughput.

One solution to the problem is to use a much lower pruning threshold than 0.1. However, using a very low threshold (or in the worst case not pruning any FNs at all) can lead to too many FNs. Instead, we used the following approach, which favors MORE: for each scenario, we repeated the simulation for different values of the pruning threshold α, starting with the default value of 0.1, and lowering it by 0.01 until no receiver was disconnected. This last value was the one we used as the pruning threshold in the comparison against *Pacifier*. For the 10 scenarios studied, the largest pruning threshold that does not cause any disconnection varies from 0.1 to 0.03.

2) Impact of tree-based opportunistic routing: We start the evaluation of *Pacifier* by examining the impact of its tree-based opportunistic routing, by comparing the basic version of *Pacifier* (TREE), with MORE. The only difference between the protocols is the algorithm used for selecting FNs and assigning TX_credits to them. The results for 10 different scenarios are shown in Figure 2.

Figure 2(a) shows TREE achieves higher throughput than MORE in 8 out of 10 scenarios. The gain ranges from 20% (Scenario 7) up to 199% (Scenario 4), with an average throughput gain over all 10 scenarios equal to 42%. Only in two scenarios (2 and 3), there is a small throughput reduction with TREE, about 16%.

The higher throughput achieved by TREE compared to MORE can be explained by the fewer FNs and lower total number of transmissions in the former compared to the latter. In particular, Figure 2(b) shows that the use of a tree instead of a union of belts reduces the number of FNs in TREE by 36% on average, compared to MORE. Figure 2(c) shows the use of a tree combined with the new algorithm for TX_credit calculation results in on average 44% reduction in the total number of transmissions in TREE, compared to MORE. Finally, Figure 2(d) shows MORE has a high source redundancy; the source sends on average 17 times the file size. TREE reduces the average source redundancy to 12. The difference in source redundancy suggests TREE is more efficient in selecting FNs and more accurate in calculating the TX_credit values for the FNs.

3) Impact of source rate limiting: We next evaluate the impact of backpressure-based rate limiting at the source, as implemented in the TREE+RL version of *Pacifier*. Figure 3(a) shows that the use of rate limiting at the source improves the throughput by 5% (Scenario 6) to 94% (Scenario 1), with an average of 20%, compared to TREE. Figure 3(c) shows that TREE+RL on averages reduces the source redundancy to 5.84, a 52% reduction compared to the value of 12.15 for TREE. The reduction in the source redundancy in turn reduces the total number of transmissions by 28% on average, as shown in Figure 3(b). We found that this reduction comes not only from the contribution of the source but also from the majority of the FNs. This confirms that, by pacing the source's transmissions, the source's children and grandchildren get better chances to successfully transmit packets and make progress down the tree.

4) Solving the "crying baby" problem: The above results have shown that TREE and TREE+RL already offer significant throughput improvement over MORE. However, these two versions of *Pacifier* still suffer from the "crying baby" problem. We next evaluate the effectiveness of round-robin batching on solving the "crying baby" problem, by comparing TREE+RL+RRB (the complete *Pacifier* protocol) with TREE+RL.

Figure 4 shows the average throughput achieved with TREE+RL+RRB and TREE+RL in each of the 10 scenar-

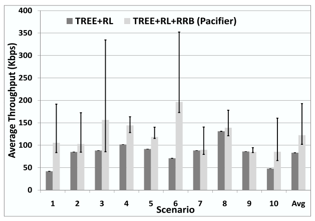


Fig. 4. Throughput with TREE+RL and TREE+RL+RRB (*Pacifier*) for 10 different scenarios. The error bars show throughput of the best and the worst receiver.

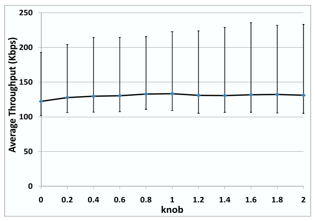
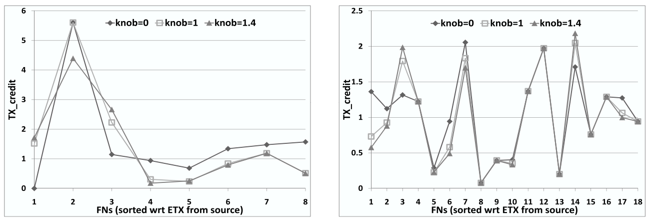


Fig. 5. Average throughput with *Pacifier* as a function of $knob$, over 10 scenarios. The error bars show average max and min values over the 10 scenarios.



(a) FN TX_credits - Scenario 3.

(b) FN TX_credits - Scenario 10.

Fig. 6. FN TX_credits with 3 different $knob$ values in 2 scenarios. FNs are sorted in increasing ETX distance from the source.

ios, as well as the throughput of the best and the worst receiver (top and bottom of error bars) in each scenario under TREE+RL+RRB. We make three observations. First, with TREE+RL, which uses sequential batch transmission, all 9 receivers in each scenario achieve the same throughput, which is determined by the worst receiver. In contrast, with TREE+RL+RRB, well-connected receivers get much higher throughput than the average, as shown by the large gap between the top of the error bars and the average in most scenarios. Averaging over 10 scenarios, the best receiver achieves 58% higher throughput than the average throughput by all receivers. Second, allowing receivers to proceed independently in TREE+RL+RRB also increases the average throughput by 47% on average over all 10 scenarios, compared to TREE+RL. Third, importantly, the throughput improvement for the best receivers comes at almost no penalty to the worst receivers. In particular, compared to with TREE+RL, the throughput of the worst receiver with TREE+RL+RRB gets slightly worse in 3 scenarios (Scenario 7, 8, and 9 by 10%, 7%, and 3%, respectively), remains unaffected in 2 scenarios (Scenarios 2 and 3), and increases by 26%-146% for the remaining 5 scenarios.

In summary, *Pacifier* not only solves the “crying baby” problem, by allowing well-connected receivers to proceed fast, but at the same moment it also makes the “crying baby” itself (i.e., the worst receiver) “happier” in the majority of the cases. This is because as more and more receivers complete decoding, the source can prune the tree branches towards them, reducing the number of FNs and the amount of contention in the network.

5) Tuning the knob in TX_credit Calculation: Finally, we study the intricacies in calculating TX_credit values by varying the $knob$ value introduced in Section III-C2. We vary the value of $knob$ from 0 (the version evaluated in Section IV-B4) to 2. Intuitively, as $knob$ increases, the throughput of the best receiver is expected to decrease and the throughput of the worst receiver is expected to increase, since we spend more time on each batch in every round.

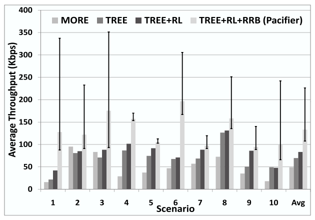
Figure 5 shows the average, max, and min throughput with *Pacifier*, as $knob$ varies from 0 to 2. Every point is the average over 10 scenarios. Somewhat surprisingly, higher $knob$ values improve the max throughput and $knob = 1$ appears to maximize the average and the min throughput. $knob = 0$, which is expected to optimize the performance of the best receiver, achieves on average the lowest max, average, and min throughput, compared to all the other $knob$ values. This confirms our speculation in Section III-C that setting $knob = 0$ may not give the best result as the TX_credit calculation is fundamentally based on the expected opportunistic receptions, and a lower than expected number of receptions in any given batch can cause the best receiver to be a few packets short of decoding a batch and wait for a whole round.

An additional counter-intuitive observation from Figure 5 is that the throughput does not change monotonically as the $knob$ increases. The reason for this behavior is that the the TX_credits assigned to FNs actually interfere in a very complex way. In a nutshell, increasing the TX_credit of an FN j can potentially decrease the TX_credit of its child nodes, as the grandchild nodes of j now have more chance of overhearing j’s transmissions. Consequently, the chance of packet reception at j’s grand-grandchild nodes from their upstream nodes is affected in complicated ways.[8] As an example, Figures 6(a)- 6(b) plot the TX_credit values of the source and FNs in the sorted order (based on ETX values from the source) for Scenarios 3 and 10 for $knob$ values of 0, 1.0, and 1.4. We observe changing the $knob$ value almost always increases the TX_credit for some FNs while decreasing the TX_credit for some other FNs.

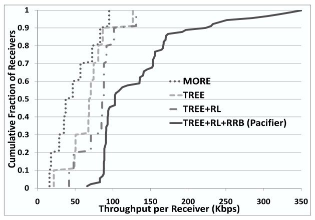
In summary, the discussion above shows that there is no optimal value for $knob$. We find setting $knob = 1$ in *Pacifier* appears to improve the max throughput while maximizing the average and the min throughput.

6) Overall Comparison: Figure 7(a) summarizes the average, maximum and minimum throughput comparison between MORE, TREE, TREE+RL, and TREE+RL+RRB (*Pacifier*), where TREE+RL+RRB used a $knob$ value of 1. We observe that on average, *Pacifier* outperforms TREE+RL, TREE, and MORE by 60%, 90%, and 171%, respectively. In addition,

[8]Recall our TX_credit calculation is a polynomial heuristic; optimal TX_credit assignment to all FNs is an NP-hard problem.



(a) Average, max, and min throughput with each protocol for each of the 10 scenarios.



(b) CDF of the 90 throughput measurements obtained with each protocol for 10 scenarios with 9 receivers each.

Fig. 7. Overall throughput comparison of MORE, TREE, TREE+RL, and TREE+RL+RRB (*Pacifier*).

Pacifier allows well-connected receivers to achieve much higher throughput, which can be up to 20x higher than with MORE (for scenario 1), and also improves throughput of the worst receiver in all 10 scenarios, compared to the other 3 protocols.

Figure 7(b) depicts the same results in a different way. It plots the CDF of the 90 throughput values obtained from 10 scenarios with 9 receivers each, for the four protocols. In this figure, the CDFs for MORE, TREE, and TREE+RL have a staircase form, since for each scenario, all 9 receivers get roughly the same throughput (equal to that of the worst receiver) due to the “crying baby” problem. In contrast, with *Pacifier*, receivers finish independently of each other and the CDF has a continuous form. In the median case, *Pacifier* outperforms TREE+RL, TREE, and MORE by 20%, 49%, and 178%, respectively.

The benefit of *Pacifier* becomes more prominent if we look at the two ends of the CDF. *Pacifier* solves the “crying baby” problem by allowing good receivers to achieve very high throughput. The 90th percentile is 223Kbps for *Pacifier*, 70%, higher than with TREE+RL, 77% higher than with TREE, and 159% higher than with MORE. If we look at the 10th percentile, i.e., the worst receivers, we observe that *Pacifier* outperforms TREE+RL, TREE, and MORE by 80%, 300%, and 450%, respectively. This shows again that *Pacifier* not only solves the “crying baby” problem, it also simultaneously offers a significant improvement to the performance of the “crying baby” itself.

V. Conclusion

In this paper, we presented *Pacifier*, the first practical NC-based high-throughput, reliable multicast protocol for WMNs. *Pacifier* seamlessly integrates tree-based opportunistic routing, intra-flow NC, source rate limiting, and round-robin batching, to support high-throughput, reliable multicast routing, while at the same time it offers a simple yet very efficient solution to the “crying baby” problem. Extensive simulations showed that *Pacifier* increases the average throughput gain over the state-of-the-art MORE by 171%, while the maximum throughput gain for well-connected receivers can as high as 20x.

Acknowledgment

This work was supported in part by NSF grant CNS-0626703.

References

[1] “MIT Roofnet,” http://www.pdos.lcs.mit.edu/roofnet.
[2] “Bay area wireless users group,” http://www.bawug.org.
[3] “Seattle wireless,” http://www.seattlewireless.net.
[4] S. Biswas and R. Morris, “ExOR: Opportunistic multi-hop routing for wireless networks”, in *Proc. of ACM SIGCOMM*, 2005.
[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, “XORs in the air: Practical wireless network coding”, in *Proc. of ACM SIGCOMM*, August 2006.
[6] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, “Trading structure for randomness in wireless opportunistic routing”, in *Proc. of ACM SIGCOMM*, 2007.
[7] S. Katti, S. Gollakota, and D. Katabi, “Embracing wireless interference: Analog network coding”, in *Proc. of ACM SIGCOMM*, 2007.
[8] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, “A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks”, in *Proc. of WPMC*, 2004.
[9] R. Chandra, V. Ramasubramaniam, and K. Birman, “Anonymous gossip: Improving multicast reliability in mobile ad hoc networks”, in *Proc. of IEEE ICDCS*, 2001.
[10] E. Pagani and G. Rossi, “Reliable broadcast in mobile multihop packet networks”, in *Proc. of ACM MOBICOM*, 1997.
[11] A. Sobeih, H. Baraka, and A. Fahmy, “ReMhoc: A reliable multicast protocol for wireless mobile multihop ad hoc networks”, in *Proc. of IEEE CCNC*, 2004.
[12] V. Rajendran, Y. Yi, K. Obraczka, S.-J. Lee, K. Tang, and M. Gerla, “Combining source- and localized recovery to achieve reliable multicast in multi-hop ad hoc networks”, in *Proc. of IFIP Networking*, 2004.
[13] B. Scheuermann, M. Transier, C. L. M. Mauve, and W. Effelsberg, “Backpressure multicast congestion control in mobile ad-hoc networks”, in *Proc. of ACM CoNEXT*, 2007.
[14] D. Koutsonikolas and Y. C. Hu, “The case for FEC-based reliable multicast in wireless mesh networks,” in *Proc. of IEEE/IFIP DSN*, 2007.
[15] L. Rizzo and L. Visicano, “RMDP: an FEC-based reliable multicast protocol for wireless environments”, *Mobile Computing and Communications Review*, vol. 2, no. 2, 1998.
[16] D. Lun, M. Medard, and R. Koetter, “Efficient operation of wireless packet networks using network coding”, in *Proc. of IWCT*, 2005.
[17] M. Ghaderi, D. Towsley, and J. Kurose, “Reliability Gain of Network Coding in Lossy Wireless Networks”, in *Proc. of IEEE INFOCOM*, 2008.
[18] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton, “Log-based receiver-reliable multicast for distributed interactive simulation”, in *Proc. of ACM SIGCOMM*, 1995.
[19] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, “A reliable framework for light-weight sessions and application level framing”, *IEEE/ACM Transactions on Networking*, 1997.
[20] L. Rizzo, “Effective erasure codes for reliable computer communication protocols”, *ACM Comp. Comm. Review*, vol. 27, no. 2, 1997.
[21] J. Nonnenmacher, E. Biersack, and D. Towsley, “Parity-based loss recovery for reliable multicast transmission”, in *Proc. of ACM SIGCOMM*, 1997.
[22] E. Schooler and J. Gemmel, “Using multicast FEC to solve the midnight madness problem”, Technical Report, MSR-TR-97-25, 1997.
[23] M. Luby, “LT codes,” in *Proc. of 43rd FoCS*, 2002.
[24] E. Vollset and P. Ezhilchelvan, “A survey of reliable broadcast protocols for mobile ad-hoc networks”, University of Newcastle upon Tyne, Technical Report, CS-TR-792, 2003.
[25] J. Luo, P. Eugster, and J.-P. Hubaux, “Route Driven Gossip: Probabilistic reliable multicast in ad hoc networks”, in *Proc. of IEEE INFOCOM*, 2003.
[26] J. Sang Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, “Codecast: a network-coding-based ad hoc multicast protocol”, *IEEE Wireless Communications*, vol. 13, no. 5, 2006.
[27] D. S. J. D. Couto, D. Aguayo, J. C. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing”, in *Proc. of ACM MOBICOM*, 2003.

[28] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu, “Simple opportunistic routing protocol for wireless mesh networks”, in *Proc. of IEEE WiMesh*, 2006.
[29] S.-J. Lee, M. Gerla, and C.-C. Chiang, “On-Demand Multicast Routing Protocol”, in *Proc. of IEEE WCNC*, September 1999.
[30] C. Gkantsidis, W. Hu, P. Key, B. Radunovic, S. Gheorghiu, and P. Rodriguez, “Multipath code casting for wireless mesh networks”, in *Proc. of ACM CoNEXT*, 2007.
[31] 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*, 2006.
[32] X. Zhang and B. Li, “Optimized multipath network coding in lossy wireless networks”, in *Proc. of IEEE ICDCS*, 2008.
[33] B. Scheuermann, C. Lochert, and M. Mauve, “Implicit hop-by-hop congestion control in wireless multihop networks”, *Elsevier Ad Hoc Networks*, vol. 6, no. 2, pp. 260–286, Apr. 2008.
[34] X. Zeng, R. Bagrodia, and M. Gerla, “Glomosim: A library for parallel simulation of large-scale wireless networks”, in *Proc. of PADS Workshop*, May 1998.