# WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks

Rabin Patra* rkpatra@cs.berkeley.edu
Sergiu Nedevschi* sergiu@cs.berkeley.edu
Sonesh Surana* sonesh@cs.berkeley.edu
Anmol Sheth† anmol.sheth@coloardo.edu
Lakshminarayanan Subramanian‡ lakshmi@cs.nyu.edu
Eric Brewer* brewer@cs.berkeley.edu

*WiFi-based Long Distance (WiLD) networks with links as long as 50–100 km have the potential to provide connectivity at substantially lower costs than traditional approaches. However, real-world deployments of such networks yield very poor end-to-end performance due to two reasons. First, the current 802.11 MAC protocol has fundamental shortcomings when used over long-distances. Second, WiLD networks can exhibit high and variable loss characteristics, thereby severely limiting end-to-end throughput.*

*This paper describes the design, implementation and evaluation of WiLDNet, a system that overcomes these two problems and provides enhanced end-to-end performance in WiLD networks. To address the protocol shortcomings, WiLDNet makes several essential changes to the 802.11 MAC protocol, but continues to rely on standard WiFi network cards. To better handle losses and improve link utilization, WiLDNet uses an adaptive loss-recovery mechanism using FEC and bulk acknowledgements. Based on a real-world deployment, WiLDNet provides a 2–5 fold improvement in TCP/UDP throughput (along with significantly reduced loss-rates) in comparison to the best throughput achievable by conventional 802.11 MAC. WiLDNet can also be configured to adapt to a range of end-to-end performance requirements (bandwidth, delay, loss, jitter).*

## 1 Introduction

Many developing regions around the world, especially in rural and remote areas, require low-cost network connectivity solutions. Traditional approaches based on telephone, cellular, satellite or fibers have proved to be an expensive proposition especially in low population density and low-income regions. In Africa, even though cellular and satellite coverage is available in rural regions, bandwidth is extremely expensive due primarily to low user densities (satellite usage cost is about US$3000 per Mbps per month) [18]. WiMax [27], another proposed solution, is currently also very expensive and has been primarily intended for carriers (like cellular). WiMax is hard to deploy in the “grass roots” style typical for developing regions.

WiFi-based Long Distance (WiLD) networks [8, 9] are emerging as a low-cost connectivity solution and are increasingly being deployed in developing regions. The primary cost gains arise from the use of very high-volume off-the-shelf 802.11 wireless cards, of which over 140M were made in 2005. These links exploit unlicensed spectrum, and are low power and lightweight, leading to additional cost savings [6]. These networks are very different from the short-range multihop urban mesh networks [5]. Unlike mesh networks which use omnidirectional antennas to cater to short ranges (less than 1–2 km at most), WiLD networks comprise of point-to-point wireless links that use high-gain directional antennas (e.g. 24 dBi, 8 degree beam-width) to focus the wireless signal (for line of sight) over long distances (10–100 km).

Despite the promise of WiLD networks as a low-cost network connectivity solution, our experience has shown that the performance of WiLD networks in real-world deployments is abysmal. The main reasons for this poor performance are two-fold. First, the stock 802.11 protocol has fundamental *protocol shortcomings* that make it ill-suited for WiLD environments. Three specific shortcomings include: (a) the *802.11 link-level recovery* mechanism results in low utilization; (b) *inappropriateness of CSMA/CA* over long distances; (c) WiLD networks experience *inter-link interference* which introduces the need for synchronizing packet transmissions at each node [20]. The second problem is that the links in our WiLD network deployments (in US, India, Ghana) experienced very *high and variable packet loss-rates* induced by external factors (primarily external WiFi interference in our deployment); under such high loss conditions, TCP flows hardly progress and continuously experience timeouts.

In this paper, we describe the design and implementation of WiLDNet, a system that addresses all the aforementioned problems and provides enhanced end-to-end perfor-

*Univ of California, Berkeley
†Univ of Colorado, Boulder
‡New York University

mance in multi-hop WiLD networks. Prior to our study, the only work addressing this problem was 2P [20], a MAC protocol enhancement proposed by Raman *et al.*. 2P addresses primarily inter-link interference, and proposes a TDMA-style protocol with synchronous node transmissions. The design of WiLDNet leverages and builds on top of 2P, making additional changes to the 802.11 standard in order to further improve utilization and to make the system robust in the face of packet loss. The key factors that distinguish WiLDNet from 2P and the stock 802.11 protocol are four-fold:

*1. Improving link utilization using bulk packet acknowledgments:* The current 802.11 protocol uses a stop-and-wait link recovery mechanism, which when used over long distances with high round-trip times leads to under-utilization of the channel. To improve link utilization, WiLDNet uses a bulk packet acknowledgment protocol.

*2. Designing TDMA in lossy environments:* The stock 802.11 CSMA/CA mechanism is inappropriate for WiLD settings since it cannot assess the state of the channel at the receiver. 2P proposed a basic TDMA mechanism (instead of CSMA/CA) that explicitly synchronized transmissions at each node to prevent inter-link interference. However, in the face of packet losses (especially high-loss rates), explicit synchronization can lead to deadlock scenarios due to loss of synchronization marker packets. In WiLDNet, to deal with this problem, we resort to an implicit approach, using loose time synchronization between nodes to determine a TDMA schedule that is not affected by packet loss.

*3. Handling high packet loss-rates:* In our WiLD network deployments, we found that external WiFi interference is the primary source of packet loss. The emergence of many WiFi deployments, even in developing regions, will exacerbate this problem. In WiLDNet, we use an adaptive loss-recovery mechanism that uses a combination of FEC and bulk acknowledgments to reduce significantly the perceived loss rate and to increase the end-to-end throughput. Here, we show that WiLDNet's link-layer recovery mechanism is much more efficient in comparison to higher layer recovery mechanisms like Snoop.

*4. Application-based parameter configuration:* Different applications have varying end-to-end requirements in terms of bandwidth, loss, delay and jitter. In WiLDNet, configuring the TDMA and loss recovery parameters (time slot period, FEC, number of retries) provides a trade-off spectrum across different end-to-end properties. We explore this trade-off spectrum and show that WiLDNet parameters can be configured to suit different end-to-end application requirements.

We have implemented all our modifications as a *shim layer* above the driver using the Click modular router [14]. We have deployed WiLDNet in our campus testbed comprising of 5 long-distance wireless links. Figure 1 shows the topology of our campus testbed. Apart from the design and implementation of WiLDNet, we have had two years

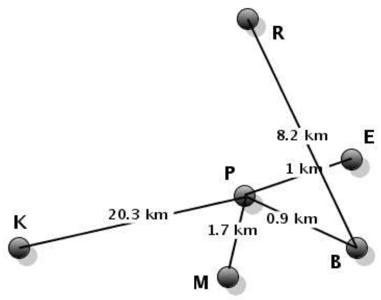


Figure 1: Overview of the WiLD campus testbed (distances not to scale)

experience in deploying and maintaining two production WiLD networks in India and Ghana that support real users. Our network at the Aravind Eye Hospital, India, provides interactive patient-doctor video-conferencing services between the hospital and five surrounding villages (10–25 km away from the hospital). It is currently being used for about 1700 remote patient examinations per month. The design of WiLDNet that is presented in this paper has continuously evolved in the past two years to solve many of the performance problems that we faced in our deployments.

Using a detailed performance evaluation, we roughly observe a 2–5 fold improvement in the TCP throughput over WiLDNet in comparison to the best achievable TCP throughput obtained by making minor driver changes to the conventional 802.11 MAC across a wide variety of settings. In the multi-hop case with bidirectional traffic under lossy channel conditions, our system provides a sustained TCP/UDP throughput of 5 Mbps, which is 2.5 times larger than the maximum throughput achievable using conventional 802.11b. The bandwidth overhead of our loss-recovery mechanisms are minimal. In the near future we intend to transition our system from the campus testbed into the two production networks in India and Ghana.

## 2 WiLD performance issues

In this section, we describe in detail two important causes for poor end-to-end performance in WiLD settings: (a) 802.11 protocol shortcomings; (b) high and variable loss-rates in the underlying channel induced by external factors. We begin by providing a brief description of WiLD networks in Section 2.1.In Section 2.3, we elaborate on three protocol shortcomings of 802.11 in WiLD settings: (a) CSMA/CA; (b) link-level recovery; (c) inter-link interference. For each of these, we show that manipulating the driver level parameters is insufficient to achieve good performance over long-distance links. Finally in Section 2.4, we describe the loss characteristics of our deployed WiLD networks. In our testbed, we observed the primary cause of these losses to be external WiFi interference and not multipath effects. Finally, in Section 2.5, we discuss the effect of these two causes on TCP performance.

## 2.1 WiLD networks: an introduction

The IEEE 802.11 standard (WiFi) was designed for wireless broadcast environments with many hosts in close vicinity competing for channel access. Wireless radios are half-duplex and cannot listen while transmitting; consequently, a CSMA/CA (carrier-sense multiple-access/collision avoidance) mechanism is used to reduce collisions. Unlike standard WiFi networks, WiFi-based Long Distance (WiLD) networks use multihop point-to-point links, where each link can be as long as 80–100 km. To achieve long distances in single point-to-point links, nodes use directional antennas with gains as high as 30dBi, and may also use high-power wireless cards with up to 400mW of transmit power output. Additionally, in multihop settings, nodes have multiple radios with one radio per fixed point-to-point link to each neighbor. Each radio can operate on different channels if required. This is different from standard 802.11 networks where all nodes route traffic through an access point and contend for the medium on a single channel. Some real life deployments of WiLD networks include the Akshaya network [26], the Digital Gangetic Plains project [4], and the CRCnet project [8]. The Akshaya network is one of the largest deployments in the world with over 400 nodes and links going up to 30 kms.

## 2.2 Experimental Setup

We use three different experimental setups for conducting our measumements and for evaluation of the implementation of WildNet.

**Campus testbed**: Figure 1 is our real-world campus testbed on which we have currently deployed WiLDNet. The campus testbed consists of links ranging from 1 to 44km km. Also the end-points in our campus testbed are located in areas with varying levels of external WiFi interference. **Wireless Channel Emulator**: The channel emulator (Spirent 5500 [24]) enables repeatable experiments by keeping the link conditions stable for the duration of the experiment. Moreover, by introducing specific propagation delays we can vary the length of the link to distances longer than any of the links currently available in our campus testbed and can hence study the effect of long propagation delay. We can also study this in isolation of external interference by placing the end host radios in RF isolation boxes and connecting them to the emulator through RF cables.

**Indoor multi-hop testbed**: We also perform controlled multi-hop experiments on an indoor multi-hop testbed consisting of 2 hops.. We can also control the amount external interference in the RF isolation boxes by placing an additional wireless node in a isolation box just to transmit packets mimicking a real interferer. The amount of interference is controlled by the rate of the CBR traffic from this node. This setup has a very small propagation delay; we use it only to perform experiments evaluating TDMA scheduling and loss recovery from interference.

We use Atheros 802.11 a/b/g radios for all our experiments. The wireless nodes are 266 MHz x86 Geode single board computers running Linux 2.4.26. We use iperf to measure UDP and TCP throughput. The madwifi Atheros driver was modified to collect relevant PHY and MAC layer information.

## 2.3 802.11 protocol shortcomings

In this section, we study three main limitations of the 802.11 protocol: the inefficient link-layer recovery mechanism, collisions in long distance links, and inter-link interference. These limitations make 802.11 ill-suited even in the case of a single WiLD link. Based on extensive experiments, we also show that modifying the driver-level parameters of 802.11 is insufficient to achieve good performance.

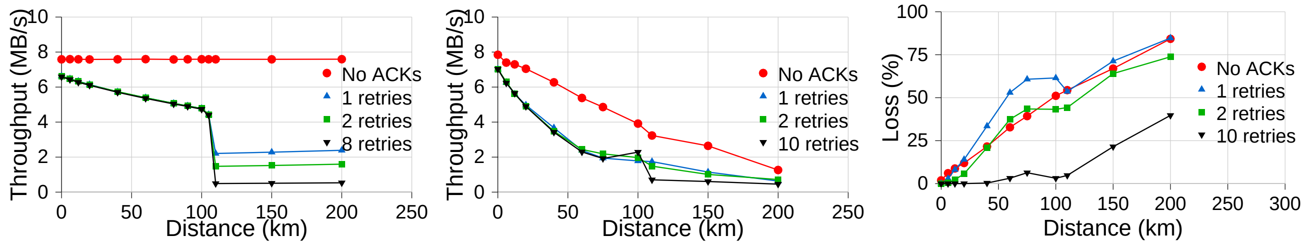
### 2.3.1 Inefficient link-layer recovery mechanism

The 802.11 MAC uses a simple stop-and-wait protocol, with each packet independently acknowledged. Upon successfully receiving a packet, the receiver node is required to send an acknowledgement within a tight time bound (ACK-Timeout), or the sender has to retransmit. This mechanism has two drawbacks:

• As the link distance increases, propagation delay increases as well, and the sender waits for a longer time for the ACK to return. This decreases channel utilization.

• If the time it takes for the ACK to return exceeds the ACKTimeout parameter, the sender will retransmit unnecessarily and waste bandwidth.

We illustrate these problems by performing experiments using the wireless channel emulator. To emulate long distances, we configure the emulator to introduce a delay to emulate links ranging from 0–200 km. Figure 2(a) shows the performance of the 802.11 stop-and-wait link recovery mechanism over increasing link distance. With the MAC-layer ACKs turned off (No ACKs), we achieve a throughput of 7.6 Mbps at the PHY layer data rate of 11 Mbps. When MAC ACKs are enabled, we adjust the ACK timeout as the distance increases. In this case, the sender waits for an ACK after each transmission, and we observe decreasing channel utilization as the propagation delay increases. At 110 km, the propagation delay exceeds the maximum ACK timeout (746$\mu$s for Atheros, this is smaller for Prism 2.5 chipsets and cannot be modified) and the sender always times out before the ACKs can arrive. We notice a sharp decrease in received throughput, as the sender retries to send the packet repeatedly (even though the packets were most likely received), until the maximum number of retries is reached (this happens because, if an ACK is late, it is ignored). This causes the received throughput to stabilize at $BW_{110km}/(no\_of\_retries + 1)$.



(a) Unidirectional UDP throughput (b) Bidirectional UDP throughput (c) Bidirectional UDP loss

Figure 2: UDP with adjusted ACK timeouts with Atheros cards. Traffic is 1440 byte UDP packets in 802.11b at PHY layer datarate of 11Mbps

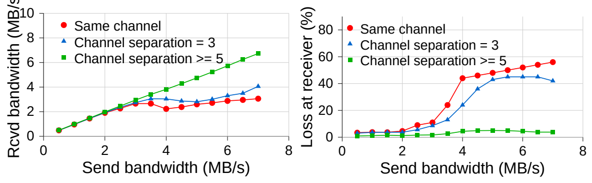
### 2.3.2 Collisions on long-distance links

The 802.11 protocol uses a CSMA/CA channel-access mechanism, in which all nodes listen to the medium for a specified time period (DIFS) before transmitting a packet to ensure that the channel is idle. This translates to a maximum allowable distance at which collisions can be avoided of about 15km for 802.11b (DIFS is 50$\mu$s), 10.2 kms for 802.11a and 8.4km for 802.11g. However for longer links, it is possible that a node would start transmitting a packet unaware of another packet transmission at the other end. As the propagation delay increases, this probability of loss from collisions increases.

We illustrate the above shortcoming using a simple experiment: we send bidirectional UDP traffic at the maximum possible sending rate on the emulated link and measure the percentage of packets successfully received at each end. Figure 2(c) shows how the packet loss rate increases with distance. Figure 2(b) shows the sum of the throughputs achieved at both ends for bidirectional UDP traffic as we increase the distance for a link. Note that there are no losses due to attenuation or outside interference in this controlled experiment; all of the losses are due to collisions.

A possible solution to this issue is to increase the DIFS time interval to account for longer propagation delays. However just as in the case of the the ACK timeout, this would decrease channel utilization substantially for longer links. Also, we not aware of any wireless chipsets that allow the DIFS interval to be configured.

### 2.3.3 Multiple Link Interference



(a) UDP throughput received (b) UDP loss at receiver

Figure 3: Effect of interference on received throughput and error rate when sending through a relay node. Separation is no. of channels in 802.11b. Traffic is 1440 byte UDP packets in 802.11b at PHY layer datarate of 11Mbps

Another important source of errors is the interference between adjacent 802.11 links operating in the same or overlapping channels. Two such adjacent links can operate without interference by using non-overlapping channels. However, Raman et al. [20] present numerous reasons why in many cases it is advantageous to operate such links on the same frequency channel. Moreover, there are WiLD topologies such as the Akshaya network [26] where different channels cannot be allocated to all the pairs of adjacent links, given the high connectivity degree of several nodes.

Inter-link interference occurs because the high-power radios create a strong RF field in the vicinity of the radio enough to interfere with the receptions at nearby radios. Directional antennas also have sufficiently high gain (4–8dBi) side lobes [4] in addition to the primary main lobes.

The first problem is when multiple radios on a node try to transmit at same time. As soon as one radio starts transmitting after sensing the carrier to be idle, all other radios in the vicinity find the carrier to be busy and backoff. This is desirable in a broadcast network to avoid collisions between two senders at any receiver node. However, in our network where each of these radios transmits over point-to-point long distance link to indepedent receivers, this backoff leads to suboptimal throughput. A second problem occurs when packets being received at one link collide with packets simultaneously transmitted on some other link on the same node. The signal strength of packets transmitted locally on a node overwhelmes any packet reception on other radios.

In order to illustrate these effects, we perform experiments on the real-world setup presented in Figure 1. First, we try to transmit UDP packets to both node K and M from P simultaneously. The total send throughput on both links is 14.20 Mbps when they are on non-overlapping channels (separation $\geq$ 4) but drops to only 7.88 Mbps when on the same channel. Next we send UDP packets from node M to node K, relayed through node P at different transmitting rates. We then measure received throughput and packet loss rate for various channel spacing between the two adjacent links, as presented in Figures 3(a) and 3(b). We observe that interference does reduce the utilization of the individual links and significantly increases the link loss-rate (even in the case of partially overlapping channels).

Therefore, the maximum channel diversity that one can simultaneously use at a single node in the case of 802.11(b) is restricted to 3 (channels 1,6,11) which may not be sufficient for many WiLD networks. This motivates the need for synchronizing both transmit and receive packet transmissions across adjacent links to improve throughput.

## 2.4 Channel Induced Loss

Apart from protocol shortcomings, another cause for poor performance is high packet loss-rates in the underlying channel due to external factors. We refer to these as *channel induced losses*. In this section, we will briefly summarize the relevant conclusions from the study carried out by Sheth *et al.* [23] where they conduct a detailed analysis of loss characterisation on our WiLD network deployments.

**Loss magnitude and variability**: Figure 4 illustrates the loss variation across time on three different links in our testbed. We find that the loss is highly varying with time and there are bursts of high loss of lengths varying from few milliseconds upto several minutes. However on the urban links, there is always a non-zero residual that varies between 1–10%. The residual loss rates in our rural links are negligible. Finally, we found the loss characteristics along a single link to be highly asymmetric. One example is illustrated in Figure 4 where we observe a negligible loss-rate from P to K but a very high loss-rate from P to K.

**Sources of loss**: The study (Sheth *et al.* [23]) investigates two potential sources that trigger channel losses in WiLD links: external WiFi interference and multipath interference. It finds that external WiFi interference to be the dominant source of packet loss and multipath to have much lesser effect.

Multipath has a very small effect because the delay spreads in WiLD environments are an order of magnitude lower than that of mesh networks. This is because as link distance increases, the path delay difference between the primary line-of-sight (LOS) path and secondary reflected paths becomes small enough to avoid inter-symbol interference(ISI). On the other hand, the primary path signal can be significantly attenuated from the secondary paths that undergo a phase shift of 180○ after reflection. This is also verified in our study [23] where we see that all our long-

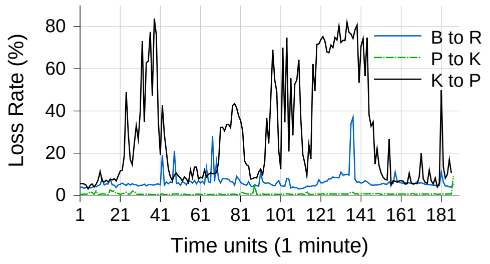


Figure 4: Packet loss variation over a period of about 3 hours. Traffic was 1Mbps CBR UDP packets of 1440 bytes each at PHY datarate of 11Mbps in 802.11b

distance links in rural areas have very low loss. In comparison, an urban mesh-network deployment (like Roofnet) has shorter and many non-line-of-sight links and thus loss from ISI is a much bigger problem.

However if WiLD network is deployed in the presence of external interfering sources, the hidden terminal problem can be much worse than in the case of an urban mesh network (with omnidirectional antennas). Due to the highly directional nature of the transmission, a larger fraction of interfering sources within range of the receiver act as hidden terminals since they cannot sense the transmission. In addition, due to long propagation delays, even external interfering sources within the range of a directional transmitter can interfere by detecting the conflict too late. Measurements on our outdoor testbed links and indoor testbed show that there is strong correlation between loss and volume of traffic from external sources on the same or adjacent channels [23].

This is unlike the case of mesh networks like Roofnet [1] where the authors conclude that multipath interference was a significant source of packet loss.

**Other factors**: Measurements on our testbed show that there is no measurable non-wifi interference in our urban links [23]. This is indicated by the absence of significant correlation between noise floor (reported by the wireless card) and loss rates. Also, the loss rates on different channels are not correlated to each other implying the absence of any wide-band interfering noise. Experiments with different 802.11 PHY datarates showed that smaller the datarates can have higher loss rates in some conditions. This can be explained by the fact that packets at lower datarates take longer time on air air and are thus more likely to collide with external traffic. Other studies by Raman *et al.* [7] show that weather conditions doesn't have noticable effects on loss rates in long distance links.

## 2.5 Impact on TCP

Taken together, protocol shortcomings of 802.11 and channel induced losses significantly lower end-to-end TCP performance. The use of stop-and-wait over long distances reduces channel utilization. In addition, we see correlated bursty collision losses due to interference from unsynchronized transmissions (over both single-link and multi-hop scenarios) as well as from external WiFi sources. Under these conditions, TCP flows often timeout resulting in very poor performance. To handle these losses, the only knob available in the driver is to tune the number of packet retries. Setting a higher value on the number of retries decreases the loss rate, but at the cost of lower throughput resulting from lower channel utilization.

To better understand this trade-off, we measure the aggregate throughput of TCP flows in both directions on an emulated link while varying distance and introducing a channel packet loss rate of 10%. Figure 5 presents the aggregate

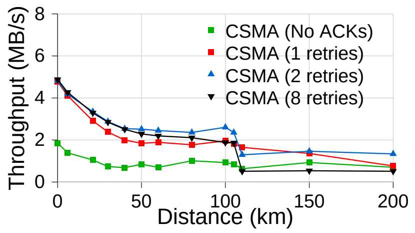


Figure 5: Bidirectional TCP over CSMA with 10% channel loss. Traffic is at PHY layer datarate of 11Mbps

TCP throughput with various number of MAC retries of the standard 802.11 MAC. Due to increased collisions and larger ACK turnaround times, throughput degrades gradually with increased distance.

## 3 WiLDNet Design

In this section, we describe the design of WiLDNet and elaborate on how it addresses the 802.11 protocol shortcomings as well as achieves good performance in the face of high-loss environments. In the previous section, we identified three basic problems with 802.11; (a) low utilization, (b) breakdown of CSMA/CA, and (c) inter-link interference. To address the problem of low utilization, we propose the use of bulk packet acknowledgments (Section 3.1). To address the limitation of CSMA and the inter-link interference problem, we build upon the TDMA protocol design of 2P [20], and make the necessary changes to adapt 2P to high-loss environments (Section 3.2). Additionally, to handle the challenge of high and variable packet losses, we design adaptive loss recovery mechanisms that use a combination of FEC and retransmissions with bulk acknowledgments (Section 3.3).

WiLDNet follows three main design principles. First, the system should not be narrowly focused to a single set of application types. It should be configurable to provide a broad trade-off spectrum across different end-to-end properties like delay, bandwidth, loss, reliability and jitter. Second, all mechanisms proposed should be implementable on commodity off the shelf 802.11 cards. And third, the design should be lightweight, such that it can be implemented on the resource constrained single board computers (266 MHz CPU and 128 MB memory) used in our testbed.

### 3.1 Bulk Acknowledgments

We begin with the simple case of a single WiLD link, with each node having a half duplex radio. In this case, CSMA/CA over long distances is not capable of assessing the state of the channel at the receiver. Given this simple case, at the minimum we require an echo protocol between the sender and the receiver that determines when each node transmits to prevent unsynchronized packet collisions (*i.e.*, both sender and receiver simultaneously transmit). In fact, the echo protocol is the simplest form of a TDMA protocol which is essential in WiLD environments [19]. Hence, from a node's perspective, we divide time into send and receive time slots.

To improve link utilization, we replace the stock 802.11 stop-and-wait protocol with a sliding-window based flow-control approach in which we transmit a *bulk acknowledgment* from the receiver for a window of packets. We generate a bulk acknowledgment as an aggregated acknowledgment for all the packets received within the previous slot. In this way, a sender can rapidly transmit a burst of packets rather than transmit each frame and wait for an acknowledgment for each.

The bulk acknowledgment can be either piggybacked on data packets (sent in the reverse direction), or sent as standalone packets if no data packets are available. By piggybacking the acknowledgments, the additional bandwidth usage is minimal . Each bulk ACK contains the sequence number of the last packet received in order and a variable-length bit vector ACK for all packets following the in-order sequence. Here, the sequence number of a packet is locally defined between a pair of end-points of a WiLD link.

Like 802.11, the bulk acknowledgment mechanism is not designed to guarantee perfect reliability. 802.11 has a maximum number of retries for every packet. Similarly, upon receiving a bulk acknowledgment, the sender can choose to advance the sliding window skipping unacknowledged packets depending on the maximum number of retries set. In practice, we can support different retry limits for packets of different flows. The bulk ACK mechanism introduces packet reordering at the link layer, which may not be acceptable for TCP traffic. To handle this, we provide in-order packet delivery at the link layer either for the entire traffic or at a per-flow level.

### 3.2 Designing TDMA in lossy environments

To address the inappropriateness of CSMA for WiLD networks, 2P [20] proposes a contention-free TDMA based channel access mechanism. 2P eliminates inter-link interference by synchronizing all the packet transmissions at a given node (along all links which operate on the same channel or adjacent overlapping channels). In 2P, a node in transmission mode simultaneously transmits on all its links for a globally known specific period, and then explicitly notifies the end of its transmission period to each of its neighbors using marker packets. A receiving node waits for the marker packets from all its neighbors before switching over to transmission mode. In the event of a loss of marker packets, a receiving node uses a timeout mechanism to switch into the transmission mode.

The design of 2P, while functional, is not well suited for lossy environments. Consider the simple example illustrated in Figure 6, where all links operate on the same or adjacent overlapping channels. Consider the case where $(X, A)$ is the link experiencing high packet loss-rate. Let

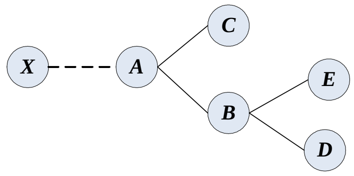


Figure 6: Example topology

$T$ denote the value of the time-slot. Whenever a marker packet transmitted by $X$ is lost, $A$ begins transmission only after a timeout period $T_0$ ($\geq T$). This, in turn, delays the next set of transmissions from nodes $B$ and $C$ to their other neighbors by a time period that equals $T_0 - T$. Unfortunately, this propagation of delay does not end here. In the time slot that follows, $D$'s transmission to its neighbors is delayed by $T_0 - T$. Hence, what we observe is that the loss of marker packets has a "ripple effect" in the entire network creating an idle period of $T_0 - T$ along every link. When multiple markers along different links are dropped, it is an open research problem to analyze the effect of simultaneous idle-period ripples propagating in the network.

Ideally, one would want $T_0 - T$ to be very small. If all nodes are perfectly time synchronized, we can set $T_0 = T$. However, in the absence of global time synchronization, one needs to set a conservative value for $T_0$. 2P chooses $T_0 = 1.25 \times T$. Hence, the loss of marker packets leads to an idle period of $0.25 \times T$ throughout the network (in 2P, this is 5 ms for $T = 20$ ms). In the face of bursty losses, transmitting multiple marker packets may not also suffice. Therefore, a single lossy link can lead to system wide inefficiencies.

Given that many of the links in our network experience sustained loss-rates over 5–40%, in WiLDNet, we use an implicit synchronization approach that aims to reduce the value of $T_0 - T$. In WiLDNet, we use a simple loose time synchronization mechanism similar to the basic linear time synchronization protocol NTP [16], where during each time slot along each link, the sender acts as the master and the receiver as the slave. Consider a link $(A, B)$ where $A$ is the sender and $B$ is the receiver at a given time. Let $t_{send\_A}$ and $t_{recv\_B}$ denote the start times of the slot as maintained by $A$ and $B$. mutually agreed upon. All the packets sent by A are timestamped with the time difference ($\delta$) between the moment the packet has been sent ($t_1$) and the beginning of the send slot( $t_{send\_A}$). When a packet is received by B at time $t_2$, the beginning of B's receiving slot is adjusted accordingly: $t_{recv\_B} = t_2 - \delta$. In practice, due to software induced variations in propagation time for different packets, the value of $\delta$ as marked in each packet may not be reflective of its true value. To handle this, we use a simple smoothing function $t_{recv\_B} := \alpha * t_{recv\_B} + (1 - \alpha) * (t_2 - \delta)$. As soon as B's receive slot is over, and $t_{send\_B} = t_{recv\_B} + T$ is reached, B starts sending for a period $T$.

Due to the propagation delay between A and B, the send and corresponding receive slots are slightly skewed. The end-effect of this loose synchronization is that the value of $T_0 - T$ is limited by the propagation delay across the link even in the face of packet losses (assuming clock speeds are roughly comparable). Hence, an implicit synchronization approach significantly reduces the value of $T_0 - T$ thereby reducing the overall number of idle periods in the network.

## 3.3 Adaptive loss recovery

Handling high and variable loss-rates primarily induced by external WiFi interference is a challenging problem. However, to achieve predictable end-to-end performance, it is essential to have an loss recovery mechanism that can hide the loss variability in the underlying channel and provide a bound on the loss-rate perceived by higher level applications along a single link. More specifically, the loss recovery mechanism should provide a loss-rate bound $q$ independent of the underlying link loss-rate.

Achieving such a bound is not easy in our setting due to two factors. First, it is hard to predict the arrival and duration of bursts; also, bursts occur very frequently in some of our links. Second, the loss distribution that we observed on our links is non-stationary even on long time scales (hourly and daily basis). Hence, it is not easy to use a simple model to capture the channel loss characteristics. In WiLDNet, we can either use retransmissions or FEC to deal with losses (or a combination of both). A retransmission based approach can achieve the loss-bound $q$ with minimal throughput overhead but at the expense of increased delay. However, our FEC approach primarily reduces the perceived loss-rate but cannot achieve arbitrarily low loss-bounds mainly due to the unpredictability of the channel. To achieve arbitrarily low loss rates using only FEC incurs a substantial throughput overhead. FEC incurs additional throughput overhead but does not incur a delay penalty especially since it is used in combination with TDMA on a per-slot basis.

### 3.3.1 Tuning the number of retransmissions

To achieve a loss bound $q$ independent of underlying channel loss rate $p(t)$, we need to tune the number of retransmissions. One can adjust the number of retransmissions $n(t)$ for a channel loss-rate $p(t)$ such that $(1 - p(t))^{n(t)} = q$. Given that our WiLD links support in-order link-layer delivery (or in-order delivery on a per-flow basis, a larger $n(t)$ also means a larger maximum delay, equal to $n(t) * T$ for a slot period $T$. One can set different values of $n(t)$ for different flows. We found that estimating $p(t)$ using an exponentially weighted average is sufficient in our links to achieve the target loss estimate $q$. A purely retransmission based recovery mechanism has minimal throughput overhead as only the lost packets are retransmitted but this comes at a cost of high delay due to the long round trip times over WiLD links.

### 3.3.2 Adaptive FEC-based recovery

Designing a good FEC mechanism in highly variable lossy conditions requires accurate estimation of the underlying channel loss. Underestimating the loss renders the throughput expended in transmitting FEC packets useless, and overestimating the loss rate leads to throughput wastage. In our environment the loss distribution is non-stationary over large time scales, making it difficult to determine an accurate loss estimator. We experimented with a variety of FEC mechanisms and found that to achieve a target loss-rate $q$ independent of the loss variation, the amount of FEC required is substantially high (often 20–40%) primarily because of the frequent occurrence of bursts.

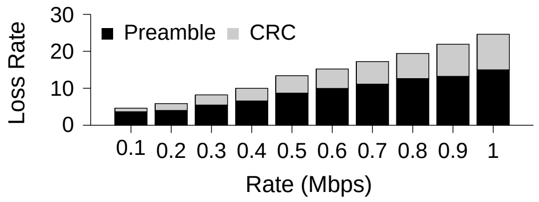


Figure 7: Breakdown of channel loss into CRC errors and preamble errors

**Motivating inter-packet FEC:** One can perform two types of FEC: inter-packet FEC (coding across packets) or intra-packet FEC (coding redundant blocks within a packet). In WiLD environments, we found that intra-packet FEC is not beneficial Based on extensive measurements on a wireless channel emulator we observe that in presence of external WiFi interference, the lost packets can be categorized into either CRC errors or preamble errors. A CRC error packet is received by the driver with a check sum error. However, an error in the preamble leads to the entire packet being dropped. This is because, the preamble has information which is used by the underlying firmware to lock onto the transmission of the receiver. Any error in the preamble would cause the firmware to drop the packet completely. Figure 7 shows the breakup of the loss rate with increasing external interference. The external interference is increased from 0.1 Mbps to 1 Mbps, and the loss rate is measured. We observe that as the external WiFi interference increases, the lost packets due to preamble errors also increase. At 1 Mbps of external interference, almost 50-80% of the lost packets are due to preamble errors. Intra-packet FEC can only handle CRC errors but cannot handle the majority of packet losses caused by preamble errors. Hence, we chose to perform only inter-packet FEC.

**Estimating the level of redundancy:** We apply FEC in combination with TDMA. For every time slot of $N$ packets, we add $N - K$ redundant packets for every $K$ packets. We use simple erasure codes based on Vandermonde matrices to generate the redundancy packets. To estimate the redundancy factor, $r = (N - K)/K$, we choose a simple but not perfect estimation policy based on a weighted average of the losses observed in the previous $M$ time slots. Here, we specifically chose a small value of $M = 10$. There are several factors that motivate this simple policy choice. First, predicting the start of a burst is very hard; hence, we do not even attempt to predict it. Second, a small value of $M$, can quickly adapt to both a start of a burst as well as reduce the FEC when a burst subsides. For a time slot of $T = 10ms$, $M = 10$ corresponds to $200$ ms to adapt to a change in the loss behavior. Third, due to non-stationary loss distributions, the added reduction that we observed in the perceived loss rate obtained by applying more complicated distribution based estimation approaches [25] is marginal. FEC is best suited for handling residual losses and long bursts. FEC is not suited to handle bursts that are shorter than the time that the weighted average FEC estimation mechanism takes to adapt.

## 4 Implementation

In this section, we describe the implementation details of WiLDNet. Our implementation comprises two parts: (a) driver-level modifications to control or disable features implemented in hardware (Section 4.1); (b) a *shim* layer that sits above the 802.11 MAC (Section 4.2) and uses the Click [14] modular router software to implement the functionalities described in Section 3.

### 4.1 Driver Modifications

The wireless cards we use in our implementation are the high power (200-400 mW) Atheros-based chipsets. WiLDNet requires the following features disabled in the driver:
**Disabling Link-Layer Association:** We disable link-layer associations in Atheros chipsets using the Adhoc-demo mode.
**Disabling Link Layer Retransmissions and Automatic ACKs:** With the Atheros drivers, we did this by using 802.11 QoS frames with WMM extensions to set the no-ACK policy.
**Disable CSMA:** We disable CSMA by turning off the Clear Channel Assessment (CCA) in Atheros chipsets using a proprietary HAL obtained from a vendor.With CCA turned off, the card can send packets without waiting for a clear channel.

### 4.2 Software Architecture Modifications

In order to implement single-link and inter-link synchronization using TDMA, the various loss recovery mechanisms, sliding window flow control, and packet reordering for in-order delivery, we use the Click Modular Router [14]. We use Click because it enables us to design a modular system with different functionalities implemented independently by various Click elements. It is also reasonably efficient for packet processing especially if loaded as

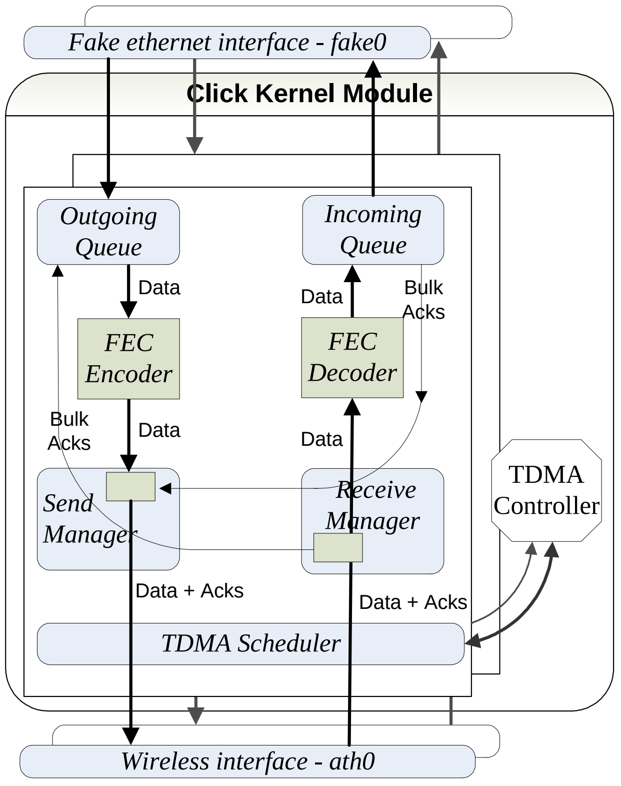


Figure 8: Click Module Data Flow

a kernel module. Click enables us to intercept and modify link-layer packets exchanged between a wireless interface and the kernel. Using kernel taps, we create fake network interfaces, such as *fake0* in Figure 8; the kernel communicates with these virtual interfaces. Click takes the packets sent to a virtual network interface, processes them, and passes them to the corresponding real wireless interface.

Figure 8 presents the structure of the Click elements of our layered system, with different functionality (and corresponding packet header processing) at various layers:
**Incoming/Outgoing Queues:** The mechanisms supporting sliding window packet flow, bulk acknowledgments, selective retransmission and reordering for in-order delivery are implemented by the incoming/outgoing queue pair. Packet buffering at the sender is necessary for retransmissions, and buffering at the receiver ensures reordering. To ensure adaptability to various application requirements, in-order delivery and packet retransmission is optional, and the maximum number of retransmission can be set on a per-packet basis.
**FEC Encoder/Decoder:** An optional layer is responsible for inter-packet forward error correction encoding and decoding. For our implementation we modify a FEC library [22] that uses erasure codes based on Vandermonde matrices computed over $GF(2^m)$. At the receiver, the reception of any $K$ out of the $N$ packets enables the recovery of the original packets. We choose this scheme because, in loss-less situations, it introduces very low latency: the original K packets can be immediately sent by the encoder (without undergoing encoding), and immediately delivered to the application by the decoder (without undergoing decoding).
**Send and Receive Managers:** These elements are responsible for managing the bulk acknowledgments and the encoded data packets. Bulk ACKs are generated by the incoming queue, piggybacked to data packets or sent as stand-alone packets (if there is no data to send), and delivered to the outgoing queue of the peer host, which uses them to delete already delivered packets.
**TDMA Scheduler:** This element ensures that packets are being sent only during the designated send slots, and manages packet timestamps as part of the synchronization mechanism.
**TDMA Controller:** This element is common for all the interfaces supported by the click module. It implements synchronization among the wireless cards on the same channel, by enforcing synchronous transmit and receive operation (all the cards on the same channel have a common send slot, followed by a common receive slot).

### 4.2.1 Timing issues

Implementing time synchronization within Click has the disadvantage of being less precise. Since there is packet queuing in the interface itself, there is variability in the time between the moment Click emits a packet and the time the packet is actually sent on the air interface. Thus, the propagation delay between the sending and the receiving click modules on the two hosts is not constant, affecting time slot calculations. Fortunately, this propagation delay is predictable for the first packet in the send slot, when the hardware interface queues are empty. Thus, in our Click implementation, we only timestamp the first packet in a slot, and use it for adjusting the receive slot at the peer. If this packet is lost, the receiver's slot is not adjusted in the current slot, but since the drift is slow this does not have a significant impact.

Another timing complication is related to estimating whether we have time to send a new packet in the current send slot. Since the packets are queued in the interface, the time when the packet leaves Click cannot be used to estimate this. To overcome this aspect, we use the notion of *virtual time*. At the beginning of a send slot, the virtual time $t_v$ is same as current (system) time $t_c$. When we send the first packet, the virtual time becomes $t_v = t_c + duration(packet)$. In general, every time we send a packet, we recompute the virtual time: $t_v = max(t_c, t_v) + duration(packet)$. And every time a packet is sent we check that the virtual time after sending this packet will not exceed the end of the send slot. If the end exceeds the end of the slot, we postpone the packet until the next slot. Although our synchronization scheme works reasonably well, we intend to move this part of the system into the interface firmware for increased accuracy.

## 5 Experimental Evaluation

The main goals of WiLDNet are to increase link utilization and to eliminate the various sources of packet loss observed in a typical multi-hop WiLD deployment, while simultaneously providing flexibility to meet different end-to-end application requirements. Raman *et al.* [20] show the improvements gained by the 2P-MAC protocol in simulation and in an indoor environment but not in a real deployment in outdoor settings. We believe these are the first actual implementation results of a protocol similar to 2P over an outdoor multi-hop WiLD network deployment.

Our evaluation has three main parts:

• We analyze the ability of WiLDNet to maintain high performance (high link utilization) over long-distance WiLD links. At long distances, we demonstrate 2–5x improvements in bidirectional TCP throughput over standard 802.11.
• Next, we evaluate the ability of WiLDNet to scale to multiple hops and eliminate inter-link interference. WiLDNet yields a 2.5x improvement in bidirectional TCP throughput on our real-world multi-hop setup.
• Finally, we evaluate the effectiveness of the two adaptive link recovery mechanisms of WiLDNet: Bulk Acks and FEC.

### 5.1 Single Link Without Channel Losses

In this section we demonstrate the ability of WiLDNet to eliminate link under-utilization and packet collisions over a single WiLD link. We compare the performance of WiLDNet with the CSMA (2 retries) base case. Unidirectional and bidirectional results with various distances (on the emulator) are presented in Figure 9 and Figure 10 respectively.

Figure 9 shows the performance of WiLDNet over a unidirectional link. The lower unidirectional throughput of WiLDNet, approximately 50% of channel capacity, is due to symmetric slot allocation between the two end points of the link. However, over longer links ($>$50 km), the TDMA-based channel allocation avoids the under-utilization of the link as experienced by CSMA. Also, beyond 110 km, CSMA begins to retransmit repeatedly after timing out waiting for Acks.

Figure 10 shows the performance of WiLDNet over a bidirectional TCP flow. In this case, WiLDNet effectively eliminates all collisions occurring in presence of bidirectional traffic. TCP throughput of 6 Mbps is maintained constant and close to the channel capacity in the bidirectional case, at increasing distances.

### 5.2 Multiple Hops

This section validates that inter-link interference is eliminated by our TDMA synchronization across links, and shows how this modification yields more than 2x TCP throughput improvements over a two-link multi-hop wireless path, assuming bidirectional TCP traffic.

| Description | Unidir | Bidir |
|---|---|---|
| Standard TCP Same Channel | **2.17** | **2.11** |
| Standard TCP Different Channels | 3.95 | 4.50 |
| WiLD TCP, Same Channel | **3.12** | **4.86** |
| WiLD TCP, Different Channels | 3.14 | 4.90 |

Table 1: Multi-hop TCP throughput comparison (in Mbps)

We illustrate this by deploying WiLDNet on our testbed shown in Figure 1. We measure TCP throughput between nodes $K$ and $M$ in two configurations: (a) operating on top of standard 802.11 MAC with maximum number of MAC retransmission and (b) operating over WiLDNet featuring inter-link synchronization. We are interested in comparing the performance of these scenarios for the case when the links operate on the same channel; however, in order to quantify the effect of interference, we also perform the same experiments with the links operating on different, non-overlapping channels, in which case the interference is almost zero, as previously shown in Figure 3.

We perform all our experiments 10 times for one minute each, and show the average results in Table 1. For bidirectional traffic, we present the sum of throughputs in each direction.

We can see that, for same channel operation, the bidirectional TCP throughput with WiLDNet (4.86 Mbps) is more than twice the throughput observed over standard 802.11 (2.11 Mbps). The improvement is substantially lower for the unidirectional case (3.14 Mbps versus 2.17 Mbps), because the WiLD links are constrained to send in one direction only roughly half of the time.

Another key observation we make is that WiLDNet is capable of eliminating almost all inter-link interference. This is shown by the fact that the throughput achieved by WiLDNet is almost the same, whether the links operate on the same channel or on non-overlapping channels. This result is very important, as it makes channel allocation a non-issue.

It is worth mentioning that the links used for these experiments are short (below 25 km); consequently, this setup isolates the improvements due to inter-link synchronization. Unfortunately, we were unable to experiment on two long (50+ km) adjacent links, which would have illustrated the combined advantage of using the long-distance modifications together with synchronization across several links.

### 5.3 WiLDNet link-recovery mechanisms

Our next set of experiments evaluate WiLDNet's adaptive link recovery mechanisms in conditions closer to the real world, where errors are generated by a combination of collisions and external interference. We evaluate the bulk ACK mechanism as well as the FEC mechanism to recover from loss.

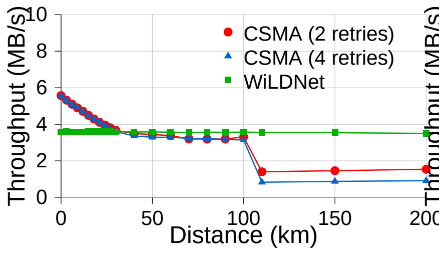


Figure 9: Unidirectional TCP

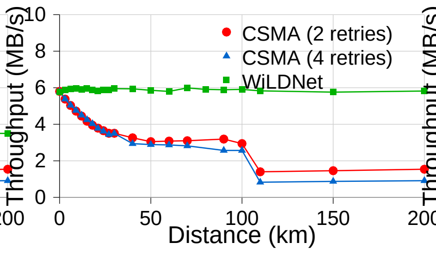


Figure 10: Bidirectional TCP

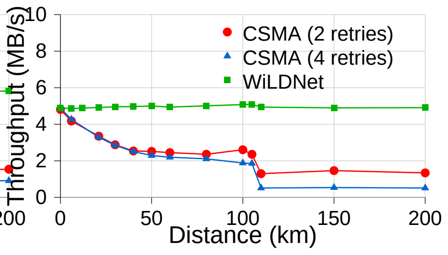


Figure 11: Bidirectional TCP with 10% channel loss rate

### 5.3.1 Bulk ACK recovery mechanism

For our first experiment, presented in Figure 11, we uniformly vary the link length on the emulator, and we introduce a 10% error rate through external interference. We again measure TCP bidirectional throughput, and compare the same variations as the ones used in the previous section. Again, WiLDNet performs the best, with throughput unaffected by distance, since the sliding-window retransmissions are not sensitive to propagation delay, as opposed to the stop-and-wait used in 802.11 CSMA. Due to the 10% error, WiLD incurs a constant throughput penalty of approximately 1 Mbps compared to the no-loss case in Figure 10.

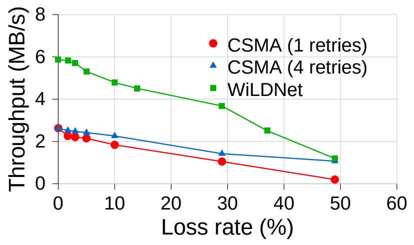


Figure 12: Bidirectional throughput for increasing loss

In our second experiment we fix the distance to 80 km, and uniformly vary external error rates. The measurement results, presented in Figure 12, show that WiLDNet maintains roughly a 2x improvement over standard CSMA with four retries, for packet loss rates up to 30%.

We also compared the performance of standard 802.11 MAC, and WiLDNet with retries enabled. We fixed the distance to 80 km, and varied the channel induced loss rates uniformly from $0$ to $50\%$. Our measurements show that WiLDNet maintains roughly a 2x improvement for packet loss rates up to 30%.

### 5.3.2 Forward Error Correction (FEC)

To measure the jitter introduced by the FEC mechanism, we performed a simple experiment where we measured the jitter of a flow under two conditions: in the absence of any loss and in the presence of a 25% loss. Figure 13 shows overhead of WiLDNet's FEC implementation. We can see that in the absence of any loss , when only encoding occurs, the jitter is minimal. However, in the presence of

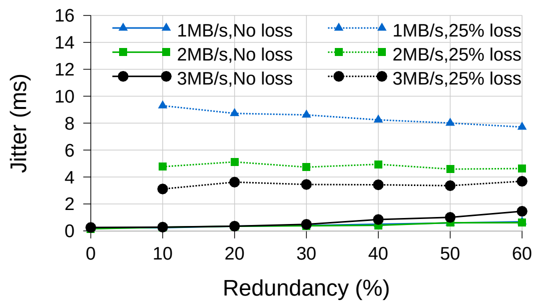


Figure 13: Overhead of the encoding and decoding process. Traffic is 1440 byte UDP CBR packets at PHY datarate of 11Mbps in 802.11b

loss, when decoding also takes place, the measured jitter increases. However, the magnitude of the jitter is very small and well within the acceptable limits of many interactive applications (voice or video), and decreases with higher throughputs (since the decoder waits less for redundant packets to arrive).

Moreover, considering the combination of FEC with TDMA, the delay overheads introduced by these methods overlap, since the slots when the host is not actively sending can be used to perform encoding without incurring any additional delay penalties.

## 6 Tradeoffs

One of the main design principles of WiLDNet is to build a system that can be configured to adapt to different application requirements. In this section we explore the tradeoff space of throughput, delay and delivered error rates by varying the slot size, number of bulk retransmissions and FEC redundancy parameters. We observe that WiLDNet can perform in a wide spectrum of the parameter space, and can easily be configured to meet specific application requirements.

### 6.1 Choosing number of retransmissions

The first tradeoff that we explore is choosing the number of retries to get a desired level of final error rate on a WiLD link. Although retransmission based loss recovery achieves optimal throughput utilization, it comes at a cost

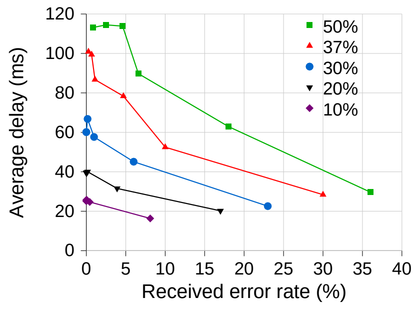


Figure 14: Avg delay with various retries

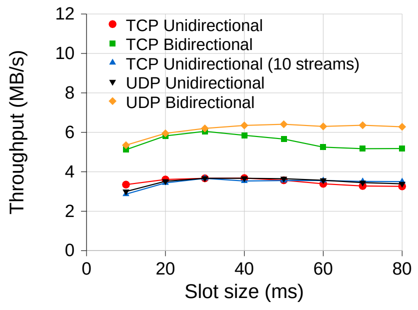


Figure 15: Throughput for various slot sizes

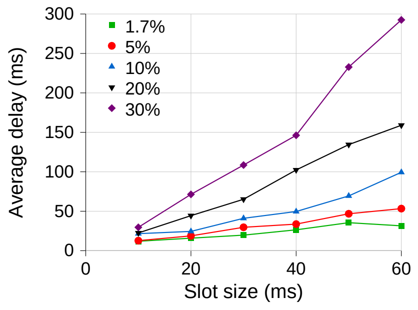


Figure 16: Avg delay for various slot sizes

of increased delay; the loss rate can be reduced to zero by arbitrarily increasing the number of retransmissions at the cost of increased delay. This tradeoff is illustrated in Figure 14 which shows a plot of delay versus error rate for varying channel loss rates (10% to 50%). Retries are increased from 0 to 10 in increments of 1 moving from right to left for a given line in the figure. All the tests are unidirectional UDP tests at 1 Mbps for a fixed slot size of 20ms. For example, at a loss rate of 50%, 3 retries are required to reduce the error rate to 5%, which leads to a delay of approximately 110 ms. We can see that as we try to reduce the final error rate at the receiver, we have to use more retries and this increases the average delay. In addition, we also observe that larger the number of retries, larger the end-to-end jitter (especially at higher loss rates).

This tradeoff has important implications for applications that are more sensitive to delay and jitter (such as real time audio and video) as compared to applications which require high reliability. For such applications, we can achieve a balance between the final error rate and the average delay by choosing an appropriate retry limit. For applications that require improved loss characteristics without incurring a delay penalty, we need to use FEC for loss recovery.

## 6.2 Choosing slot size

The second tradeoff that we explore is the effect of slot size. The two factors that affect slot size are the end-to-end delay requirements of the application and the overhead in switching from transmit to receive mode.

We first analyze the effect of slot size on received throughput. Our experiments are performed on a 60-km emulated link with 10% packet loss rate. Figure 15 presents the UDP and TCP throughputs for various slot sizes. Ideally, we would not expect the received throughput to change with slot size. However, as discussed in Section 3.2, switching between send and receive slots incurs a fixed overhead. This overhead is non-negligible for the Click-based WiLDNet implementation. Although this overhead is constant for all slot sizes, it occurs more often at smaller slots sizes. As a consequence, as seen in figure 15, at small slot sizes the achieved throughput is lower as well. However, the UDP throughput levels off beyond a slot size of 20 ms. We also observe the TCP throughput reducing slightly at higher slot size. This is because the throughput-delay product of the link increases with slot size, but the send TCP window sizes are fixed. UDP throughput is not affected at higher slot sizes.

In the next experiment, we measure the average UDP packet transmission delay while varying the slot size, for several channel error rates. The results are presented in Figure 16; each series represents a unidirectional UDP test at a particular channel loss rate. For this test a packet is retransmitted until it is acknowledged by the receiver. Figure 16 shows the increase in delay with increasing slot size. It is clear that slot sizes beyond 20 ms do not result in substantially higher throughputs, but they do result in much larger delay. Thus, if delay constraints are not too stringent, a good choice for a slot size is 20ms. However, if lower delay is required, smaller slots can be used at the expense of some throughput overhead consumed by the switching between the transmit and receive modes.

## 6.3 Choosing redundancy parameters for FEC

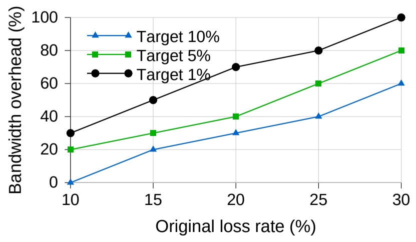


Figure 17: FEC Throughput Overhead vs Channel loss rate

The primary FEC parameter that we can tune is the redundancy factor $r = (N - K)/K$, also referred to as throughput overhead. While FEC incurs a higher throughput overhead than retransmissions, it incurs a smaller delay penalty as illustrated earlier in Section 5.3.2. To analyze the tradeoff between FEC throughput overhead and the target loss-rate, we consider the case of a single WiLD link (in our emulator environment) with a simple Bernoulli loss-model (every packet is dropped with probability $p$). Here, we set a specific value of $r$ and measure the observed target loss-rate for different values of $r$. Figure 17 shows the amount of redundancy required to meet three different target loss-rates of 10%, 5% or 1% for different error rates (namely $p$). The primary observation we make is that in order to achieve very low target-rates, FEC needs to expend a lot of throughput overhead (for example, FEC incurs a $100\%$ throughput overhead to reduce the loss-rate from $30\%$ to $1\%$).

In general, the redundancy factor required to achieve a certain target loss rate, $q$, is dependent on three factors: (a) the target loss-rate $q$; (b) the loss characteristics of the underlying channel; (c) the predictability of losses in the underlying channel. In general, when a channel is very bursty and has an unpredictable burst arrival pattern, it is very hard for FEC to achieve arbitrarily low target loss-rates. However, in such conditions, FEC can be used to deal with long bursty periods as well as steady residual loss periods to reduce the perceived loss-rate while incurring little delay penalty.

In general, for applications that can tolerate one round of retransmissions, one can perceive different combinations of FEC and retransmissions that can provide a tradeoff between overall throughput overhead, delay and target loss-rate. In the case of a channel with a stationary loss distribution, OverQoS [25] shows that the optimal policy to minimize overhead is to not use FEC in the first round but use it in the second round to pad retransmission packets. In the face of unpredictable and highly varying channel loss conditions, an alternative promising strategy is to use FEC in the first round during bursty periods to reduce the perceived loss-rate.

## 7 Related Work

**Long Distance WiFi:** The use of 802.11 for long distance networking, characterized by directional links and multiple radios per node, raises a new set of technical issues that were first illustrated in [4]. Raman *et al.*built upon this work in [20, 19] and proposed the 2P MAC protocol. WiLDNet builds upon 2P to make it robust in the face of high loss and reduce the channel under-utilization due to 802.11's stop and wait recovery mechanism. Specifically we modify 2P's implicit synchronization mechanism as well as build in two adaptive bulk ACK based and FEC based link recovery mechanisms.

**Other wireless loss recovery mechanisms:** There is a large body of research literature in wireless and wireline networks that have studied the tradeoffs between different forms of loss recovery mechanisms. Many of the classic error control mechanisms are best summarized in the book by Lin and Costello [15]. OverQoS [25] performs recovery by analyzing the FEC/ARQ tradeoff in variable channel conditions and the Vandermonde codes are used for reliable multicast in wireless environments [22].

Of particular interest for this work are the Berkeley Snoop protocol [2] which provides transport-aware link-layer recovery mechanisms in wireless environments. To compare the WiLDNet bulk ACK recovery mechanism with recovery at a higher layer, we experimented with a version of the original Snoop protocol [3] that we modified to run on WiLD links. Basically, each WiLD router ran one half of Snoop, the fixed host to mobile host part, for each each outgoing link and integrated all the Snoops on different links into one module.

We measured the performance of modified Snoop as a recovery mechanism over both standard 802.11 (CSMA) and over WiLDNet with no retries. We found that WiLDNet was still 2x better than Snoop. We also saw that Snoop was better than vanilla CSMA only at lower error rates (less than 10%). Thus, this indicates that higher layer recovery mechanisms might be better than stock 802.11 protocol, but only at lower error rates.

**Other WiFi-based MAC protocols:** Several recent efforts have focused on leveraging off-the-shelf 802.11 hardware to design new MAC protocols for different purposes. Overlay MAC Layer(OML) [21] provides a deployable approach towards implementing a TDMA style MAC on top of the 802.11 MAC using loosely-synchronized clocks to provide applications and competing nodes better control over the allocation of time-slots. SoftMAC [17] is a platform that can be used to build experimental MAC protocols. MultiMAC [10] builds on SoftMac to provide a platform where multiple MAC layers can co-exist in the networking stack and any one can be chosen on a per-packet basis.

**WiMax:** An alternative to WiLD networks is WiMax [27]. WiMax does present many strengths over a WiFi: configurable channel spectrum width (and consequently datarate), better modulation (especially for non-line of sight scenarios); operation in licensed spectrum with higher transmit power, and thus longer distances. On the other hand, WiMax currently is primarily intended for carriers (like cellular) and does not support point-to-point mode of operation. In addition, WiMax basestations are expensive ($10,000) and the high spectrum license costs in most countries dissuade grassroot style deployments. Currently it is also very difficult to obtain licenses for experimental deployment and we are not aware of open-source drivers for WiMax basestations and clients. However, most of our work in loss recovery and adaptive FEC would be equally valid for any PHY latyer (WiFi and WiMax). With appropriate modifications and cost reductions, WiMax would serve as a more suitable PHY layer for WiLD networks.

**Performance characterization:** The study carried out by Sheth *et al.* [23] on a detailed analysis of loss characterisation on our WiLD network deployments shows that loss rates in long-distance links are correlated with external interference, but multipath does not have any significant effect. The Roofnet project [1, 13] on the other hand concludes that the main source of loss in their urban mesh deployment was due to multipath interference. They see no correlation between loss rate and external WiFi interference. The authors in [7] analyse the effect of lots of factors like SNR, packet size, bitrate and weather on loss rates in long distance links. Jamieson *et al.* [12] experimentally evaluate the limitations of carrier-sense with respect to achieving high throughput in multi-hop environments. Garetto *et al.* [11] show that CSMA performs very badly in multihop wireless networks, and that this is not due to any particular implementation of a CSMA protocol, but is indeed a general coordination problem in multihop wireless networks. In this paper, we study the limitations of CSMA in WiLD network settings.

## 8 Conclusion

The commoditization of WiFi (802.11 MAC) hardware has made WiLD networks an extremely cost-effective option for providing network connectivity, especially in rural regions in developing countries. But an important stumbling block in realizing this possibility is the performance problems that these networks observe in real-world deployments. In this paper, we have attempted to bridge this gap and have identified the set of link- and MAC-layer modifications essential for achieving high throughput in multi-hop WiLD networks. Specifically, using a detailed performance evaluation, we show that the conventional 802.11 protocol is ill-suited for WiLD settings. Our proposed solution provides a 2-5x improvement in TCP throughput over the conventional MAC. Encouraged by these initial results on our long distance outdoor testbed, we will now implement these modifications in our live rural deployments in India and Ghana. We expect that these improvements can have significant impact in accelerating the penetration of feasible network connectivity options in rural regions.

## References

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

[2] H. Balakrishan. *Challenges to Reliable Data Transport over Heterogeneous Wireless Networks*. PhD thesis, University of California at Berkeley, August 1998.

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. In *ACM MOBICOM*, November 1995.

[4] P. Bhagwat, B. Raman, and D. Sanghi. Turning 802.11 Inside-out. *ACM SIGCOMM CCR*, 34:33–38, January 2004.

[5] S. Biswas and R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. *Hotnets-II*, November 2003.

[6] E. Brewer. Technology Insights for Rural Connectivity. October 2005.

[7] K. Chebrolu, B. Raman, and S. Sen. Long-Distance 802.11b Links: Performance Measurements and Experience. In *ACM MOBICOM*, 2006.

[8] Connecting Rural Communities with WiFi. http://www.crc.net.nz/index.php.

[9] Digital Gangetic Plains. http://www.iitk.ac.in/mladgp/.

[10] C. Doerr, M. Neufeld, J. Filfield, T. Weingart, D. C. Sicker, and D. Grunwald. MultiMAC - An Adaptive MAC Framework for Dynamic Radio Networking. In *IEEE DySPAN*, November 2005.

[11] M. Garetto, T. Salonidis, and E. Knightly. Modeling Per-Flow Throughput and Capturing Starvation in CSMA Multi-hop Wireless Networks. In *IEEE INFOCOM 2006*, April 2006.

[12] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the Real-World Performance of Carrier Sense. In *ACM SIGCOMM E-WIND Workshop*, August 2005.

[13] S. B. John Bicket, Daniel Aguayo and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *ACM MOBICOM*, August 2005.

[14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[15] S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[16] D. L. Mills. Internet time synchronization: The network time protocol. In *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*. 1994.

[17] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMAC - Flexible Wireless Research Platform. In *HotNets-IV*, November 2005.

[18] Partnership for Higher Education in Africa. Securing the linchpin: More bandwidth at lower cost. `http://www.foundation-partnership.org/pubs/bandwidth/index.php?chap=chap4`, 2006.

[19] B. Raman and K. Chebrolu. Revisiting MAC Design for an 802.11-based Mesh Network. In *HotNets-III*, November 2004.

[20] B. Raman and K. Chebrolu. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *ACM MOBICOM*, August 2005.

[21] A. Rao and I. Stoica. An Overlay MAC layer for 802.11 Networks. In *MOBISYS*, Seattle,WA, USA, June 2005.

[22] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, Apr. 1997.

[23] A. Sheth, S. Nedevschi, R. Patra, S. Surana, L. Subramanian, and E. Brewer. Packet Loss Characterization in WiFi-based Long Distance Networks. *IEEE INFOCOM*, 2007.

[24] Spirent Communications. http://www.spirentcom.com.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *USENIX/ACM NSDI*, March 2004.

[26] The Akshaya E-Literacy Project. http://www.akshaya.net.

[27] WiMAX forum. http://www.wimaxforum.org.