JaldiMAC – Taking the Distance Further

Yahel Ben-David[*†] Matthias Vallentin[*] Seth Fowler[*] Samuel Zats[*]

Abstract

Recently, WiFi has been hyped as an affordable technology that is capable of providing broadband Internet connectivity to poor and sparsely populated markets. A growing number of non-profit deployments, some of substantial scale[1] are making use of WiFi to extend connectivity into rural areas. However, the vast of majority of the 3.5 billion people living in rural villages[7] are still unserved.

To expand connectivity, new technology must be developed to make small rural wireless Internet service providers (WISPs) profitable. We have identified radio towers as the most expensive element that WISPs require; to reduce or eliminate this cost, we propose a novel point-to-multipoint deployment topology that takes advantage of "natural towers" such as hills and mountains to provide connectivity even over great distances. We make this design practical with a new TDMA MAC protocol called JaldiMAC[2] which *(i)* enables and is optimized for point-to-multipoint deployments, *(ii)* adapts to the asymmetry of Internet traffic, and *(iii)* provides loose quality of service guarantees for latency-sensitive traffic without compromising fairness. To our knowledge, JaldiMAC is the first integrated solution that combines all of these elements.

Our simulator-based evaluation of JaldiMAC suggests that it fulfills its design goals. Its innovative scheduler is able to provide a 71% decrease in jitter and superior latency characteristics in exchange for a 5% increase in average RX/TX switches, as compared to a standard stride scheduler. Overall, we find that JaldiMAC performs surprisingly well at this early stage.

1 Introduction

Our goal is to extend broadband Internet connectivity to currently underserved rural communities in developing nations by developing a novel technical solution. The main reasons that these communities are not currently connected are their low purchasing power and their sparse population, distributed over large distances. These factors make rural areas an unattractive market to service providers, who would be unable to recoup the required investment in infrastructure. While many researchers tend to focus on academic advancements of deployments [39] rather than its real-world applicability, we choose to maintain a conscious focus on affordability in order to customize our strategy to poor, rural communities. To this end, this paper presents a novel protocol design and a matching innovative deployment methodology. We believe that this combination offers extreme savings and will enable the proliferation of broadband Internet to rural areas in developing nations.

WiFi as the affordable choice. As with many other studies, we believe that WiFi is the most affordable and suitable technology for these target markets. Indeed, we see a growing number of Wireless Internet Service Providers (WISPs) in rural, developing countries that leverage this technology [1, 14, 2, 40]. WiFi offers notable savings in both deployment and operational costs since it does not incur spectrum licensing fees and hardware is both low-cost and available off-the-shelf. Additionally, power consumption is low, making it ideally suited for remote locations that are reliant on renewable energy sources, such as solar or wind, due to the non-availability or unreliability of grid power.

Fixed wireless vs. "blanket coverage". Unlike the setup in industrialized nations, where providers need to "blanket-cover" an area in order to service mobile stations, our setup has lower signal penetration requirements and thus, can be satisfied with highly directional antennas mounted on rooftops with a clear line of sight to the base station (typically at less then 100km away). Moreover, the use of directional antennas is attractive for its reduced interference.

Franchise replication model. Low equipment costs enable small local businesses to become WISPs while further contributing to the community by creating jobs and enhancing skills [39, 14]. The micro-franchising model could therefore be attractive for the proliferation of micro-WISP businesses. A key challenge is how to lower the entry barriers, such as the initial equipment and setup costs, for these micro-WISPs while ensuring low operational costs to drive higher profit margins and

[*]University of California, Berkeley.

[†]AirJaldi, Dharamsala, India.

[1]The AirJaldi [1] network in rural India provide Internet access to $10,000$ users in one of the world's most rural and challenging environments. The region spans an area of around 100 km radius.

[2]Jaldi is the Hindi word for fast.

ensure a faster return on investment.

Existing approaches are unattractive for rural WISP. While existing business models and technologies allow micro-WISPs to achieve financial sustainability in semi-urban areas, rural villages remain unattractive from a business point of view and consequently remain underserved. In this work, we attempt to drastically reduce the setup costs of WISPs, allowing them to serve poorer, lower-density markets while remaining profitable. We present a novel deployment topology which is much cheaper than existing alternatives. Additionally, we discuss the limitations of current protocols in the context of this topology, and address these limitations through an innovative MAC protocol.

The paper begins with a study of related work in §2 and continues in §3 with a description of our deployment methodology. After introducing the concept of different service classes in §4, we derive our architectural goals in §5. Thereafter, we present the design of JaldiMAC in §6 and evaluate it in §7. Finally, we discuss future work in §8 and conclude in §9.

2 Background and Related Work

Although 802.11 was originally designed for LAN applications, its high proliferation and low-cost drove 802.11 use beyond the designer's intentions. Both the economic attractiveness and the ability to control some of the physical and MAC-layer parameters have inspired the use of 802.11 as a research platform[27][36].

However, using 802.11 for a general-purpose wireless platform can be challenging. Its CSMA/CA MAC protocol was not intended for the outdoors and fares poorly in overcoming collisions due to the hidden node problem and the difficulty of detecting collisions at the receiver and handling backoff fairly [13]. Long distances aggravate these problems, resulting in further inefficiencies and reduced channel capacity. Despite this poor performance, 802.11's low cost has inspired many researchers to explore ways of modifying the 802.11 MAC to maximize channel utilization and allow the setup of high-bandwidth links over long distances. Most researchers suggest replacing the CSMA/CA algorithm with some type of Time Division Multiplexed Access (TDMA) in order to minimize collisions through synchronization [31, 32, 10, 37, 29, 34, 26]. The 802.11e standard was another attempt to address 802.11's shortcomings; unfortunately, 802.11e may starve traffic in some situations, and to the knowledge of the authors, it has never been fully implemented[28, 23].

2.1 Outdoor Application

Within the context of fixed outdoor applications, the field divides into three main categories: long-distance backhaul, "last mile" distribution, and mesh networks.

Long distance backhaul. 802.11 equipment is gaining a reputation of providing unmatched affordability while delivering excellent bandwidth over a very long distance. In fact, TDMA implementations such as WiLDNet [29] suggest that available bandwidth is independent of distance, with the only limitation being line-of-sight; a 382km link has been demonstrated[40]. These long-distance backhuals are ideal for linking cellular base stations and extending broadband Internet connectivity. Nevertheless, the design of WiLDNet assumes traffic is symmetric, making it very ill-suited for typical Internet applications where traffic is both highly asymmetric and dynamic. WiFiRe [31] tunes slightly better for Internet applications by assuming a 2:1 traffic ratio, but does not dynamically adapt to changing traffic patterns. WiLDNet also only allows for the creation of point-to-point links, which are impractical for the "last mile" links discussed below. The high-gain directional antennas which enable these long-distance links minimize interference as a welcome side-effect; we will take advantage of this in §3.

"Last mile" distribution. The final links which service multiple stations from a connected center are collectively referred to as the "last mile". The stations are village homes or businesses; the central location is a base station connected to a wired or wireless backhaul. Traditionally, last-mile solutions use 3 or 4 equally-spaced sector antennas mounted on a tower in the center of the village [31]. Customer Premises Equipment (CPE) would use directional antennas aimed toward the central tower. This approach was modeled after the cellular telephone industry; as we show in §3, it is an unnecessarily expensive design for sparsely populated rural markets.

Mesh networks. At the root of many academic studies lies another approach for last-mile connectivity: mesh networks [32, 26, 10, 24, 6, 4]. In a mesh, stations relay data over multiple hops; they can span long distances to offer an attractively decentralized and fault-tolerant design. Nevertheless, the cheap, widely studied single-radio mesh relays data in a store-and-forward fashion, which essentially halves the bandwidth at every hop, limits its scalability, and involves complex computation that is challenging for low-cost hardware. In a mesh, stations should be in range of two or more other stations to allow relaying and provide redundancy; this prevents the use of directional antennas, which improve signal strength, reduce interference, and allow greater spatial density. Since directional antennas are crucial for long-distance links, long-distance mesh networks are currently unrealistic, although recent work on electronically steerable

antennas [5] may eliminate this restriction. Our work must support long-distance links; however, we do borrow the concept of dynamic time slot allocation from JazzyMAC [26].

2.2 Improving Channel Utilization

Much research in this space focuses on improving channel utilization and density and compensating for the scarcity of non-overlapping channels [30, 26, 31, 18]. The optimizations in these studies improve efficiency when capacity, scale, and channel availability are an issue; in rural villages, however, this is not the case. Moreover, densely populated markets are able to utilize a growing number of solutions based on 4G technologies which may suit their requirements better than 802.11. For these reasons, we leave such considerations to future work.

2.3 Packet Loss

Sheth et al. [38] observed that rural wireless links are almost loss-free and have negligible residual loss; the main source of interference in urban networks is external WiFi networks, which are rare in rural, developing areas. The same paper notes that reducing the bitrate aggravates interference because longer transmission times increase the risk of collision; however a lower sending bitrate increases receiver sensitivity, which is attractive for weaker signals attenuated by long-distance in rural areas, as it yields better signal-to-noise ratio (SNR). Local interference can be reduced further with techniques such as the use of higher quality antennas with smaller side lobes and better front-to-back isolation ratios, proper shielding of radios and coaxial cables, grounding, relocation of the system (even on the same roof), and changing the mounting height. The effect of packet loss on TCP has been extensively studied [12, 11, 25, 42]. The problem is that TCP assumes packet loss represents network congestion, although it is a ubiquitous characteristic of wireless networks. This can be mitigated by either hiding the lossy wireless medium or using a modified TCP that differentiates between wireless loss and congestion [11]. Hiding loss better maintains the layering of the protocol stack; however, doing so aggressively can noticeably increase jitter and delay, which perturbs TCP timer calculations. The two most commonly used mechanisms to hide loss are forward error correction (FEC) and frame retransmission, also known as automatic repeat request (ARQ). Traditional 802.11 uses a per-packet ARQ scheme, but research in this field recommends FEC schemes in order to better sustain bursty losses [38, 29]. Our topology minimizes the potential for interference and loss with highly directional antennas, and rural areas exhibit low interference in any case. Therefore, we do not focus on link-layer recovery schemes in this paper and leave the integration of such mechanisms for future work.

3 Rethinking Economics and Topologies

To refine the deployment methodologies of WISPs, we examine the details of existing installations. These deployments are predominantly modeled after cellular telephony technology, which is widely used and heavily optimized. Typically, high towers are erected in the center of coverage zones (cells), and *sector antennas* are mounted on them. Sector antennas are unique in their design as they offer not only wide coverage angle (typically 90° or 120°), but also relatively high gain. As antennas are passive devices, gain in this context does not refer to amplification, but to the antenna's ability to focus energy in a particular direction; higher gain therefore implies narrower beam width. Sector antennas are exceptional because they provide relatively high gain without sacrificing beam width; unfortunately, they are very expensive. This makes them ideal for blanket coverage to mobile devices, but a poor choice for fixed wireless deployments. The sector antennas on a given tower must operate in non-overlapping frequency channels[3] to protect from mutual interference. Figure 1a shows a common sectorized deployment approach. The high gain and wide angle of sector antennas, which increases the coverage area, also makes them vulnerable to received interference from external networks or neighboring cells. In order to minimize interference, sector antennas are often mounted with a down-tilt to contain the sector's footprint. This requires higher towers if the down-tilted sectors are to provide a coverage area of reasonable size. This design is well-suited for dense urban areas where signals must penetrate walls, and the high demand in such areas makes the expense economically feasible. To improve on the conventional deployment model, it is important to understand how its costs are distributed. Towers are by far the most expensive element, without even considering real-estate costs. Power systems are the next most expensive element, and can vary substantially. Renewable energy sources are particularly attractive for remote base stations and relay towers which may receive unreliable power grid service, or no service at all; to keep the costs of such systems down, power efficiency is imperative. Antennas follow in the list of expenses, although they are substantially cheaper than the towers themselves; however, it is worth noting that sector antennas are about an order of magnitude more expensive than

[3] To cover an entire cell, three non-overlapping channels are typically needed when 120° antennas are used or four channels when 90° sectors are used.

the directional antennas used in our work.[4]. The cost of WiFi radios and routers should also be considered. To reduce the scope of the paper, we do not take labor costs into account and generally omit many other factors of the economics of wireless infrastructure.

3.1 Fixed Rural Wireless

As opposed to mobile wireless installations that support moving stations, *fixed* wireless solutions consist of directional antennas mounted on rooftops or balconies of subscriber's premises and aimed towards the base station. When applying the conventional cellular-like deployment model to rural areas, it is likely that each sector antenna would only serve a very small number of subscribers or none at all, rendering the setup prohibitively expensive. The following changes are required to the conventional deployment model to effectively support fixed rural wireless: *(i)* eliminate or substantially reduce the costs of towers, *(ii)* increase the number of subscribers per base station, which is challenging in rural settings where subscribers are sparsely distributed over large distances, *(iii)* reduce the power consumption of base stations and CPEs, and *(iv)* utilize directional instead of sector antennas to lower costs and improve resistance to interference. Eliminating the need for towers is the most desired goal; many researchers have proposed mesh networks as a potential solution to reduce the dependency upon towers. Studies on the economics of fixed rural wireless view mountains as obstacles and suggest increased costs are associated with mountainous or hilly areas [35]. Our experience working in the mountainous region of the Indian Himalayas taught us the opposite: replacing expensive towers with mountains not only proves to be cheaper, but it also extends the coverage area significantly beyond what towers can offer. Indeed, the AirJaldi [1] network in Dharamsala spans a radius of almost 100 km while using only a single modest 18 m high tower. It is also common that real estate in mountainous areas is substantially cheaper than in town centers. AirJaldi now plans and budgets deployments mostly based on the feasibility of using mountains or hills. Intuitively, one would expect most rural villages in developing countries to be situated in the plains. However, if we look beyond the typical coverage distances offered by the conventional cellular approach (typically a few hundred meters up to 2-3 km) and leapfrog to distances of 10 to 100 km, we expect to find many more villages within the coverage radius of a mountain or high hill, as depicted by Figure 1b. Low-cost directional antennas mounted on a high mountain would offer a very large coverage area, even though they have a narrow angular coverage, because the width of the coverage area increases with distance, and mountains are likely to have unobstructed line of sight to very distant locations.

[4]There may be extreme cases of very high-gain directional antennas that would be more expensive than sector antennas.

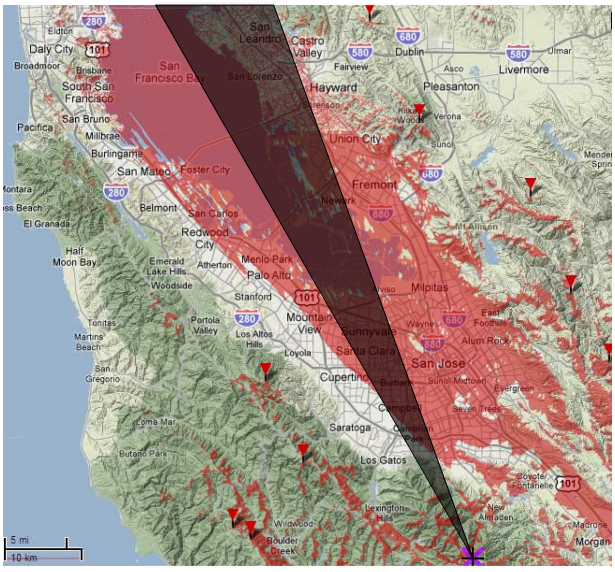


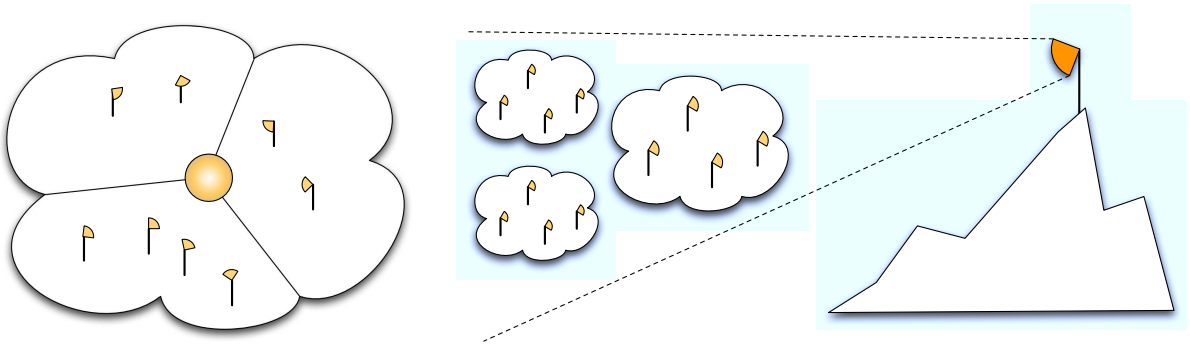
Figure 2: Coverage of the Bay Area using 11° antenna. The red areas have clear line-of-sight to the antenna on the mountain. It's interesting to note, that far locations are better covered than nearby areas which are hidden by the same mountain range on which the antenna is installed.

Figure 2 shows that a typical low-cost[5] directional antenna with a gain of 22 dBi and a horizontal beam width of 11° can easily cover much of the Bay Area if mounted on a mountain top. Far away destinations like Berkeley, Oakland, and San Leandro are well covered by a single antenna. Loma Prieta's peak is 1153 m above sea level, with an unobstructed line of sight exceeding distances of 100 km. As expected, using mountains and hills instead of towers would set new levels of coverage at a fraction of the cost.

4 Traffic Characterization

We intend our MAC protocol to support different types of service, such as VoIP, audio/video conferencing, and bulk transfers. Our choice to integrate this concept from the beginning is motivated by the difficulty of retrofitting it onto an existing MAC protocol [28, 23] and the inefficiencies of enforcing such policies at higher layers, which we discuss later. We characterize the behavior and performance of applications in the network along three dimensions: bandwidth, latency, and jitter. Bandwidth describes the application throughput over a given

[5]The current price for a 5.8 GHz 22 dBi parabolic grid antenna, beam width H10° V11°, is $47.



(a) Conventional cellular-like sectors approach.

(b) Leveraging distant elevations.

Figure 1: Conventional deployment topology vs. our approach.

period of time. Latency refers to the end-to-end delay induced by all elements on the network path. That is, latency represents the the time between the handoff of a packet to the sender's network stack and its arrival at the receiver's network stack. Jitter represents the variance of latency. We measure jitter as the average deviation from the mean latency. In general, applications often have conflicting requirements which vary the importance of these three components. This is best explained with examples: Bulk file transfers mostly care about available bandwidth and are less sensitive to latency and jitter. On the other hand, VoIP traffic needs very little bandwidth and can sustain surprisingly high latencies due to large buffers[6], but is very sensitive to jitter. The requirements of applications also vary with respect to symmetry. While applications such as VoIP and video conferencing exhibit a balanced ratio between upstream and downstream traffic, most other Internet applications use highly asymmetric traffic ratios with large downstream flows and relatively small upstream flows.

5 The Need for a New MAC protocol

Designing a holistic TDMA MAC protocol that incorporates these often conflicting aspects is a challenging undertaking. Current solutions address each aspect individually, e.g., supporting asymmetric traffic patterns [26], providing service differentiation [23], or maximizing throughput [29, 33, 31], yet none offers an integrated approach to meet the requirements in their entirety. From our discussion so far we distill a list of goals that we set out to achieve with JaldiMAC, our proposed MAC protocol:

[6] VoIP typically needs less than 16 kbps and can tolerate latencies up to 1000 ms, although user experience is better if kept below 400 ms.

1. Support **point-to-multipoint** setup over long distances.
2. Respect **dynamic traffic patterns** with varying symmetry ratios.
3. Guarantee per station **fairness**.
4. Provide strong **QoS** guarantees for different service classes.
5. Allow for **administrative preferential treatment** of stations and service classes.
6. Calibrate **PHY parameters** (bitrate, channel-width, transmit power, etc.) on the fly, based on changing physical RF conditions.
7. Handle **error correction** to hide losses from overlaying TCP traffic.

6 TDMA for Point-to-Multipoint

In this section, we present the architecture of JaldiMAC, an adaptive TDMA-based point-to-multipoint MAC protocol with support for differentiated service classes. We designed JaldiMAC to achieve the first four goals set out in §5 while remaining sufficiently flexible for future implementation of the remaining goals. After introducing common terminology in §6.1, we discuss our implicit synchronization mechanism in §6.2 which eliminates the problem of clock drift and the need for time synchronization. Next, we present our service classes in §6.3; these classes provide a mechanism for stations to request varying treatment for traffic with different requirements. The next three sections describe JaldiMAC's scheduling mechanism. First, a min-max algorithm (§6.4) assigns abstract "slots" to stations while ensuring fairness. These slots are then arranged to preserve service class guarantees by the ply scheduler (§6.5). Finally, a station mapper

transforms the slots into a final layout reflecting the stations' original requests (§6.6).

6.1 Terminology

As explained in §2, CSMA/CA is a poor fit for long-distance wireless links. This observation has sparked re-evaluation of MAC design in the research community [33, 29, 31] and industry [8, 20], shifting attention toward TDMA solutions. Different TDMA designs often use different terminology, rendering it difficult to compare their properties. In addition, the expressiveness of the models is insufficient or not generic enough. Thus, it does not allow us to illustrate the space in its entirety. Since we have not found a compelling and consistent terminology with enough expressiveness, we now define one that precisely identifies the relevant components. In Figure 3, we present a *window* of a TDMA *schedule* that partitions time into continuous time intervals. The schedule consists of *rounds*[7] that exhibit a specific structure which we term *layout*. Each round begins with a *contention slot* that allows stations to join the network or request transmission time, as discussed in §6.1.1. We define the round size as the distance between two contention slots. The other slot type is a data slot that can be used for upstream or downstream transmissions. We define a *slot* as the minimal allocatable time unit, as determined by hardware and physics constraints. Switching the sender on a per-slot basis is too fine-grained because of the overhead involved; for this reason, we will often make use of a contiguous sequence of slots assigned to the same station, which we refer to as a single *chunk*. Finally, we use the term *allocation* to describe the sum of all time a station receives in a given round.

6.1.1 Dynamic Layouts

Traditional TDMA schemes [33, 31, 29] use rounds of fixed size, with a *static* layout that changes only when stations join or leave the network. Further, the ratio between downstream and upstream traffic is usually fixed. WiFiRe, for example, uses a 2:1 ratio [31]. Even so, a hard-coded symmetry ratio and static round layouts are inherently inefficient for a link with constantly changing traffic characteristics. As the bandwidth required in each direction by each station changes, different layouts are required to ensure that the link is fully and fairly utilized. There are two general techniques used to generate these dynamic layouts:

Centralized Monitoring is a technique in which the TDMA master is exclusively responsible for changing the layout based on observed utilization. Since the intelligence is concentrated in the master, this scheme is simple to implement, but its *retrospective* nature and lack of ability to easily differentiate between types of traffic are fundamentally limiting.

[7] We avoid the commonly used term *frame* which already has specific meaning at the MAC layer.

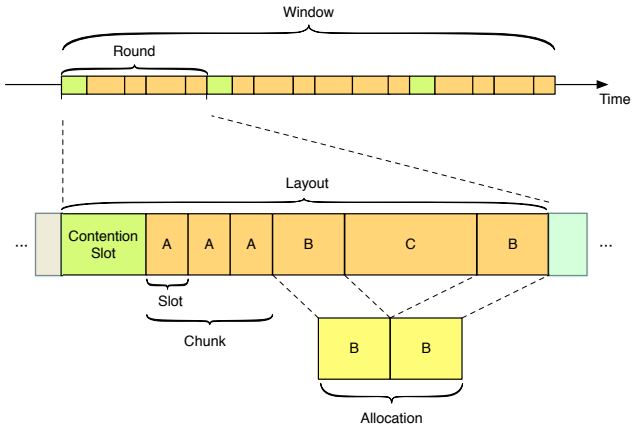


Figure 3: TDMA creates a *schedule* over time that is divided into *rounds*, each of which demarcate the distance from the beginning of two *contention slots*. The smallest allocatable unit is a *slot* and a contiguous sequence of slots form a *chunk*. The sum of all slots of a particular station represent its *allocation*.

Decentralized Reporting allows each station to individually report its anticipated traffic needs; it was pioneered for 802.11 by JazzyMAC[26]. Each station must be more sophisticated, but they can use their superior knowledge to tailor their requests to the requirements of individual traffic streams, and they're able to use their queues' current sizes and growth rates to make better predictions about the future than is possible for the master alone.

JaldiMAC uses a dynamic layout with a decentralized reporting scheme in which each of the stations requests upstream bandwidth and the master arranges the layout based upon these requests, downstream bandwidth requirements, fairness considerations, and tunable parameters that prevent excessive latency and keep the link responsive. When making their requests, stations may take into account the contents of their queues, expectations of future traffic, and the delay between the time at which they send their request and the time at which they are actually granted permission to send. The requests can be made in two ways:

Contention slot requests are used when a station did not receive any upstream slots in the previous layout. In the case a station is not given any time to transmit, it must make a bandwidth request using the contention slot. If there are many such stations, the potential for conflict is high; to compensate for

this, the contention slot is then expanded to decrease conflicts.

Follow-up requests can be made by a station that has already received slots in the current layout. As part of the data it sends in its slot, the station can make a new request.

We note that the downstream data queued for a station can be viewed as a virtual request from the master; likewise, the contention slot can be viewed as a virtual station in the schedule. We assume that this approach is used, and do not address downstream data or the contention slot specifically in the discussion that follows.

6.2 Implicit Synchronization

TDMA usually requires that the master and stations have a single understanding of the current time. If their clocks drift apart, stations begin to overlap their transmissions more and more with those of their peers, and the result is destructive interference. Because a drift of less than a millisecond can be problematic, synchronization is a challenging problem for TDMA-based MACs [29].

We propose to avoid the problem entirely by changing the structure of the layout. Rather than transmit layout updates and downstream data in one large slot, as is common in TDMA implementations, JaldiMAC's master transmits the downstream data for each station separately. At the beginning of each downstream slot, the master informs the receiving station of the length of its upstream slot. After the downstream transmission has finished, the station begins transmitting until it has either emptied its queues or the assigned slot length is over. The master then moves on to the next station in the layout, transmitting its downstream data and giving it control of the medium in turn. If a station is scheduled to transmit in the layout but has no downstream data, the master only has to transmit its upstream slot length. Similarly, if a station is not scheduled to transmit in the layout but has downstream data, the master informs the station that it has a slot length of zero.

Traditional 802.11 uses a simple ARQ protocol that operates in stop-and-wait fashion: each sent packet is acknowledged individually by the receiver, which is extremely inefficient for long distance links with high propagation delay. To solve this problem, JaldiMAC employs a *bulk acknowledgement* scheme [29] using cumulative ACKs for each station's transmission. When the master informs the next station to transmit, it piggybacks the ACK for the previous station at the beginning of its announcement. Since all stations can hear the master in our topology, the previous station can overhear the master's transmission and extract the relevant ACK information without any unnecessary RX/TX context switches.

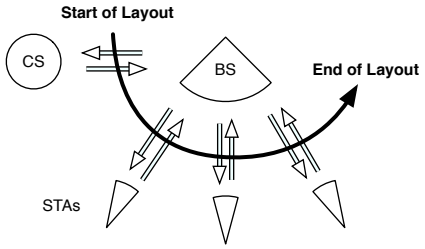


Figure 4: Layout structure for implicit TDMA synchronization. For each station, (STA) the master (BS) transmits downstream data and slot size and then receives upstream data. The contention slot (CS) is handled in the same way.

6.3 Differentiated Service Classes

The service classes we have distinguished in §4 differ based upon their demands in terms of bandwidth, latency, and jitter. (Additional dimensions, such as error correction, are also worth considering, though we leave this to future work.) JaldiMAC caters to these explicit requirements by varying each station's allocation and the frequency and variance of its chunks.

We argue that two conceptually different service models, *bulk* and *latency-sensitive*, are enough to cover the needs of the vast majority of service classes. The objective of the bulk model is to maximize throughput; latency is immaterial. Conversely, the latency-sensitive model seeks to minimize the response time and avoid jitter as much as possible.

Different latency-sensitive applications produce packets with different sizes. For example, VoIP packets are roughly 40 bytes [17] while the majority of video packets range from 1000 to 1500 bytes [15, 16]. Consequently, the minimum packet size required by the application dictates the appropriate *minimum chunk size* for the request. If we assign smaller chunks than the required packet size, the packet would have to be split across multiple chunks. This fragmentation can increase latency substantially. Similarly, a latency sensitive application generates packets at a certain rate. If the chunks that service the application's traffic are spaced more closely than the rate at which packets are generated, some of the chunks will be wasted; this will also result in an increase in latency. For these reasons, we divide the latency service model into multiple *latency service classes*, each with a different minimum chunk size and different *period*.

We attempt to guarantee that the size and period properties of latency-sensitive service classes will be honored in the final round layout; however, certain combinations of these classes cannot be honored simultaneously.

Admittedly, our algorithms inherently introduce some rounding because of the quantization the intermediate steps require. Therefore, while JaldiMAC's enforcement of these properties is effective, it is also imperfect. We describe JaldiMAC as offering "loose guarantees" for its service classes.

JaldiMAC is distinguished by two other atypical design decisions. Many TDMA designs require chunks to be allocated in multiples of a fixed-size slot; JaldiMAC only requires that the chunks be at least as large as the minimum chunk size. To provide more consistent latency and jitter properties over time, we apply an Exponentially Weighted Moving Average (EWMA) to the period of latency-sensitive requests. Due to time constraints, we were unable to fully implement the former, and we could not implement the latter at all; therefore, we leave evaluation of these features to future work.

Having sketched design principles, we now turn to the actual algorithms that JaldiMAC employs. We begin by discussing our mechanism for ensuring fairness between stations.

6.4 Allocation Fairness

In the context of point-to-multipoint TDMA, transmission time is the bottleneck resource; *fairness* in this environment is the equitable sharing of time between multiple stations. Existing point-to-multipoint TDMA schemes [26, 9] use an equality-oriented notion of fairness known as Jain's fairness index [21], which they enforce by bounding transmission time. Unfortunately, it becomes problematic to apply this measure when stations can voluntarily yield if they finish early. Moreover, for our design, it is more suitable to define fairness relative to the station's requested time. Consequently, we use the *max-min* fairness criterion instead.

In max-min fairness, no station receives more time than it has requested and excess capacity is split evenly among the stations with outstanding demand. For example, when the available time is 9 and the station requests are $\vec{\rho} = (2, 4, 5)$, max-min assigns each stations its fair share $9/3 = 3$ according to its needs: $(2, 3, 3)$. This assignment leaves an excess of 1 which is evenly split among the remaining stations, resulting in the final assignment of $(2, 3.5, 3.5)$. If fractional assignments are not possible, as in our implementation, uneven excess shares are assigned arbitrarily.

We employ max-min fairness to treat each station equally with respect to the sum of all its requests across classes. In other words, we ensure fairness among stations without considering which traffic classes each station has chosen for its requests.

The use of max-min fairness with our latency-sensitive traffic model is complicated by the fact that a latency-sensitive request does not have a fixed size. Instead, its size in a given round depends upon the number of chunks it is allocated in that round, which in turn depends upon its period and its first chunk's position in the layout. For example, a latency-sensitive request with a period of 5 slots will appear twice in a 10-slot round if its first chunk is placed in the first slot, but only once if it starts at the sixth slot. This problem is aggravated by the fact that JaldiMAC allows round sizes to vary. We have addressed this problem using worst-case analysis: we compute the size of a latency-sensitive request by assuming the maximum round size and that the request's first chunk starts at the beginning of the round. As expected, this method can result in an overestimate of the request's true size and an overallocation to the node in rounds where there is relatively little contention. The result is that the scheduler will leave holes in the layout which it expects to be filled by the request, but which the request cannot actually fill. A correct solution to this problem would make use of knowledge about the scheduling algorithm in use to avoid overestimating; because we wanted to be able to experiment with scheduling algorithms freely, we did not use this approach. Instead, at the station mapping stage described in §6.6, we dynamically allocate the unused slots to any nodes which did not receive their entire request.

In our implementation, we begin by aggregating all requests from each station which are in the same service class; all bulk requests reside in one class, while the latency-sensitive requests are grouped by period and size. These aggregate requests are then reformulated in terms of slots, rather than bytes. The max-min fairness algorithm takes an input of the total number of slots requested by each station and yields fairly distributed allocations of each station. Each station's allocation is then divided proportionally among the service classes it has requested, with the restrictions that no class receives zero slots, if possible, or more slots than its requests require. These per-class allocations are then sent to the next stage in our design: the ply scheduler.

6.5 Ply Scheduling

To turn the per-class allocations for each node into a round layout, we use ply scheduling, a variant of stride scheduling[41]. Stride scheduling, originally invented to schedule processes, distributes access to a resource between requests in a periodic fashion. Stride scheduling ensures that requests receive access proportional to their priorities. It also minimizes variance and prevents starvation. In the case of TDMA, we are dealing with stations instead of processes, and transmission time (or equivalently, transmission size) instead of CPU time, but stride scheduling is still an attractive algorithm because of these properties.

Unfortunately, JaldiMAC has multiple dimensions of

priority. We are not only concerned with the total transmission time allocated to a station, but also with the size of the chunks that transmission time is broken into, and the period of time between them. With traditional stride scheduling, there is no direct control over chunk size, and the total transmission time for a station is inherently linked to the period of time between chunks - the size of the allocation for a station can only be expressed in stride scheduling by controlling how often that station will be scheduled in the layout.

We address this problem by enhancing stride scheduling to support the minimum chunk sizes discussed in §6.3. In our modified stride scheduling algorithm, the service classes' periods are used as strides to select a class to place; the class is then allocated a number of slots equal to its minimum chunk size. The bulk class, which has no minimum chunk size or period, is assigned 1 for both values. We additionally ensure that no class receives more slots than it has requested; this produces a layout that approximates the ideal behavior for each class. Unfortunately, minimum chunk sizes seriously interfere with the stride scheduler's ability to minimize jitter, as we show in §7; our experiments led us to believe that the problem could not be solved without a fundamental redesign of the algorithm.

Our solution was *ply scheduling*, a new algorithm that improves upon stride scheduling by placing each service class separately, rather than assigning each slot in the layout in sequential order as stride scheduling does. We place latency-sensitive service classes before the bulk class, because we want to ensure that the actual periods we deliver for those classes are as close as possible to those requested by station, since the latency the traffic experiences will be kept to a minimum if the station is able to transmit as soon as the next packet is ready. Even more important, we want the periods to be consistent, so that jitter is minimized. Service classes with a larger minimum chunk size rapidly become more difficult to place as more of the layout becomes full, so we place these before classes with smaller minimum chunk sizes.

Each "ply" is a virtual layout that is equal in size to the total number of unassigned slots in the preceding ply. The scheduler starts by placing the highest priority service class (the latency-sensitive service class with the largest minimum chunk size, if any latency-sensitive requests have been made) on an initial ply equal in size to the full round; it assigns slots to that class, spaced according to the class's period. The empty slots that remain unassigned then determine the size of the next ply. The service class which has the next highest priority is then placed in this ply in the same way, and the process continues until the lowest priority service class (generally bulk) has been placed. This process is illustrated in Figure 5. Each ply's assignments are then mapped onto the

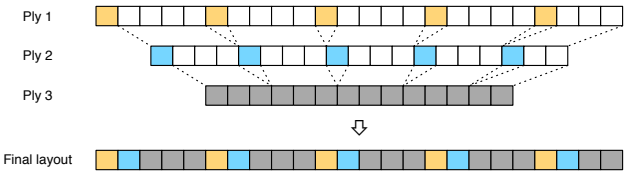


Figure 5: A "ply" is a virtual layout with a size equal to the number of unassigned slots in the preceding ply.

empty slots in the preceding ply, until the first ply has been reached and the final layout has been determined. The relationship betweens plies is illustrated in Figure 5; pseudocode for the ply algorithm is provided in the appendix.

After the ply scheduler has completed its task, each service class has been assigned space in the layout equal to the size of its allocation. To create the final layout, JaldiMAC breaks down the ply scheduler's output further by assigning the slots to individual stations, and fine-tuning the layout at the byte (rather than slot) level. This task is performed by our station mapper.

6.6 Station Mapping

Once a layout has been constructed in terms of service class slots, the next step is to assign these slots to individual stations based upon their requests. For the latency sensitive service classes, JaldiMAC's approach is to assign slots in a round-robin fashion to each station with a matching request. Since a service class has a single minimum chunk size for all requests, a normal stride scheduler can be used to carry out this assignment.

Bulk requests can be placed with more freedom; JaldiMAC takes advantage of this to minimize RX/TX switches and the overhead they entail. The station mapper's strategy is to place slots for the same station next to each other; to do so, it identifies unassigned intervals, which are sequences of bulk-class slots that have not yet been assigned to a station. First, the station mapper places "seeds" for any stations which have only bulk requests, since they are not yet represented in the layout. Each seed is placed in the middle of the largest remaining unassigned interval. From this point, the station mapper follows an iterative process: for each bulk request which has still not received all of the slots it was allocated, the station mapper finds the largest unassigned interval which has a slot for the same station on one end, and assigns a new slot for that station next to the already-assigned one. If no appropriate intervals were found, the station mapper places a new seed for that station. The process continues until every bulk request is satisfied.

This method alone yields a suboptimal layout. The problem is that a seed can sometimes be placed in a way that divides what could have been a single large chunk for a station into two smaller chunks, with a chunk

for another station in the middle. To solve this problem, JaldiMAC examines each interval in the layout that consists only of bulk requests and rearranges it so that chunks from a given station are placed together; this can be done with any sorting algorithm. The stations assigned to the latency-sensitive slots that book-end the interval are then identified, and any chunks in the interval that belong to those stations are moved to the appropriate end of the interval. This process minimizes RX/TX switches within the constraints imposed by rest of the layout. We note that, due to time constraints, our simulator does not fully implement this optimization.

To construct the final layout for the current round, JaldiMAC converts this slot-based layout into one based upon bytes. The entire round is represented as a stream of bytes, and the chunks for each station are converted into intervals in this stream. Bulk request intervals are greedily filled until all of the station's bulk requests are satisfied. Latency-sensitive request intervals are sized according to the size parameter of their service class; because JaldiMAC rounds these values up to a whole number of slots for layout purposes, the resulting intervals will sometimes not fill the entire time dedicated to that slot. Due to time constraints, we do not take advantage of these gaps; however, it would be possible to arrange the layout so that the gaps can be used to give more time to stations which have additional data to transmit. We believe that this is a crucial optimization to ensure that the medium is used as efficiently as possible, but we leave the algorithm for future work.

7 Evaluation

In the previous section, we described JaldiMAC, our novel point-to-multipoint TDMA MAC with differentiated service classes, max-min fair allocation, and dynamic round layout. As JaldiMAC is unique in its combination of features, we now try to understand and characterize its performance from various perspectives.

We first evaluate the scheduling component of JaldiMAC in §7.1, where we compare our ply scheduler against stride scheduling [41]. Next, we shift our focus in §7.2 to the granularity of stations and investigate the node mapping component. After assessing the context switch overhead in §7.3 we summarize our observations in §7.4.

7.1 Class-level Jitter

In the following, we compare our ply scheduler with the previously mentioned stride [41] scheduler, which is a deterministic proportional-share scheduling technique with low response time variance. Considered from a networking perspective, stride scheduling represents resource rights by tickets that reflect the throughput of the stations. The frequency (or latency) is directly proportional to the amount of allocated tickets. That is, a station with half as many tickets as another is scheduled half as often.

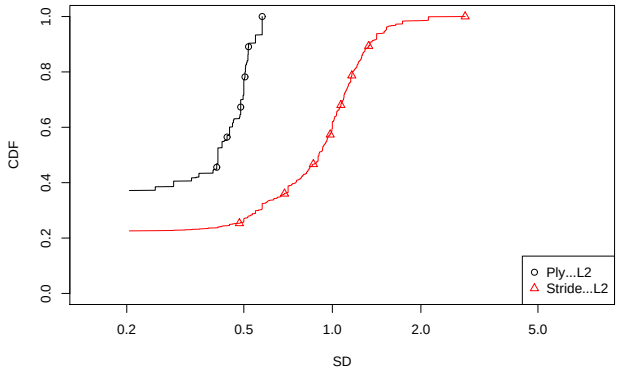
This comparison is at the level of service classes, i.e., we construct a layout for classes rather than stations. In a later step, stations' requests are mapped onto this layout (see §7.2). For this evaluation, we fix the round size and generate all possible allocations for three different service classes: bulk, latency L_1, and latency L_2. Since we make no guarantees with respect to jitter for the bulk class, we omit it. The only difference between L_1 and L_2 is the minimum chunk size, which is 1 for L_1 and 2 for L_2. (We anticipate these to be the most common in practice.) Figure 6 displays our results for a round size of 50 slots. Jitter is plotted as the standard deviation of the inter-chunk distance for each service class. As we can see in Figure 6a, ply exhibits remarkably lower jitter values (median = 0.41) compared to stride (median = 0.90) for service class L_2. Ply still performs better in service class L_1 with a median 0.84, compared to the median 0.97 of stride, although with a longer tail that indicates poor performance in a small number of pathological cases. It is worth noting that a realistic minimal time slot size for 802.11g hardware is 5ms[26], which would give this round a durations of 250ms. In practice, 250ms is our desired upper limit, as we would like to remain below the resulting 500ms round trip latency.

The superior performance of ply can be explained by juxtaposing the details of the algorithms' operations. Ply places the largest latency-sensitive class *entirely* before moving on to the next smaller one. Stride works its way incrementally through the round, placing the "next" class as determined by each class' stride and the placements that have already occurred. The linear and local nature of this approach results in higher deviation from the optimum when dealing with classes with varying minimum chunk sizes. This is due to the fact that the algorithm is so sensitive to the sizes of the chunks that have already been placed. Ply, however, achieves placement at or close to the theoretical optimum for high-priority classes by placing them first, rendering it particularly attractive for scenarios that require loose service guarantees.

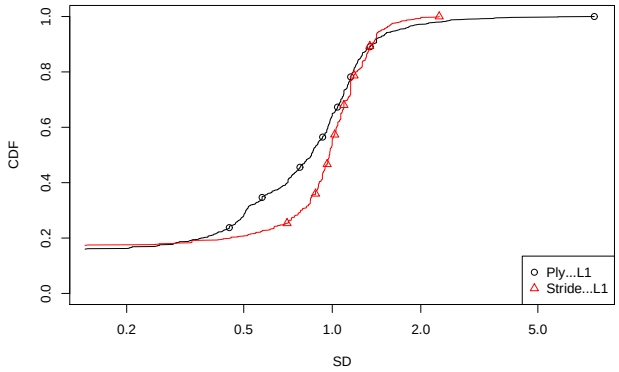
7.2 Request-level Jitter and Latency

The above jitter analysis profiles ply and stride on a service class level, which covers only a portion of the sequence of action. After placing the service classes, the individual stations' granted requests must be mapped onto the resulting layout. Ultimately, this is where jitter matters the most since the actual stations, and not the classes, will suffer from high jitter.

For this evaluation, we generated 10,000 random sets of requests. Each set contained random requests for five nodes. To simplify the analysis, the requests for



(a) Roundsize 50 (250ms), latency class L_2.



(b) Roundsize 50 (250ms), latency class L_1.

Figure 6: Comparing jitter of ply and stride over all possible allocations. The x-axis (log scale) shows the standard deviation of the period between two successive chunks for the given service classes. The y-axis shows the Cumulative Distribution Function (CDF).

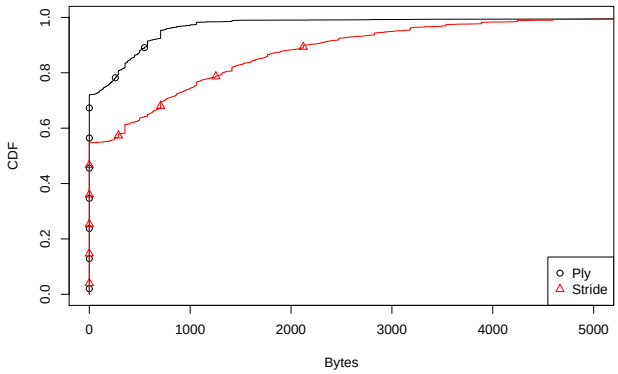


Figure 7: Jitter at the request level, analyzed in terms of the variance of inter-chunk periods for each latency-sensitive request.

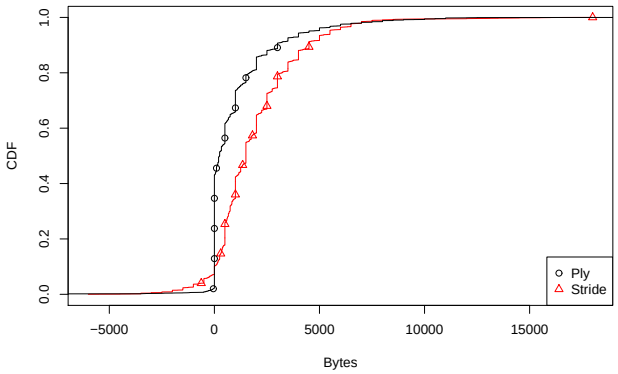


Figure 8: Difference between average period and requested period.

each node are constrained to contain no more than one latency-sensitive service class; an arbitrary number of bulk requests are allowed. (This still means JaldiMAC must contend with up to five different latency-sensitive service classes for each set of requests.) The slot size is 500 bytes, and the round size is 50 slots; each bulk request may be up to 10,000 bytes, while each latency-sensitive request's class may have a maximum size of 1,000 bytes and a maximum period of 25,000 bytes. These requests are run through a simulator that performs all of the tasks described in §6, from max-min fairness to station mapping. We ran the same sets of requests through our simulated implementation twice, once using the stride scheduler to generate the layout, and once using the ply scheduler.

Our first experiment examined the jitter experienced at the individual request level; as before, we only consider latency-sensitive requests. In Figure 7, we see that both stride and ply perform quite well, with a median jitter of 0 bytes. However, the average jitter of stride (671 bytes) is almost 248% higher than the average jitter of ply (193 bytes).

In our second experiment, we compare the difference between the average period that is actually allocated for each request versus the requested period. We round the requested period up to the next slot size to focus on the differences between the algorithms, as neither algorithm can handle periods at a finer resolution than this. As shown in Figure 8, ply is able to provide the best possi-

ble period at the slot level for about 50% of the requests, and its performance remains better or equal to stride in all cases. Any increase in the period above what was requested means that the scheduling algorithm has introduced unnecessary latency; ply's better ability to deliver the requested period will therefore be observed as lower latency by applications. Additionally, stride scheduling sometimes delivers a period that is too short; this is made visible in the figure by the portion of its CDF that has a negative value. In this situation, the latency-sensitive application may not yet have delivered the next packet, so the slot may go to waste. Ply always avoids this problem, which is a highly desirable property.

7.3 RX/TX Context Switches

Having discussed jitter and latency, we evaluate overhead introduced by JaldiMAC. As mentioned in §6.2, long distance wireless links suffer from overhead associated with RX/TX context switches. At the same time, our loose service guarantees require a timely preemption of the schedule to minimize jitter, which inherently requires more context switches.

Figure 9 quantifies the number of RX/TX switches for a given round. For this analysis, we fed our simulator 10,000 rounds with a mean and median round size of 20,860 and 24,000 respectively. Our comparison of stride and ply in Figure 9a shows that there is barely a difference between the two. Stride has 14.52 context switches on average, whereas ply follows closely with 15.26. This difference is negligible in contrast to the dramatic difference in the jitter comparison in §7.1.

We are also interested in the number of RX/TX context switches as a function of round size, which is depicted by Figure 9b. Here, we see that the number of context switches only increases slowly with the number of rounds. To visualize the relationship between round size and RX/TX switches, we add a Loess Curve which uses local point sets to create a function which describes the data variation.

7.4 Results

Overall, we find that JaldiMAC performs surprisingly well at this early simulated stage.

There exists a clear trade-off between achieving our loose service class guarantees and RX/TX context switches. We currently only have a simulated implementation of JaldiMAC, and a real implementation would allow us to make a better cost-benefit analysis using real traffic. Nevertheless, our preliminary data suggests that the increase in RX/TX switches caused by using ply scheduling is very small, and is likely worth the cost to benefit from its superior ability to deliver low jitter and latency to sensitive applications.

8 Future Work

In this section, we present future work. Due to time limitations, we were not able to fully implement every aspect of JaldiMAC in our simulator; these short-term issues are noted in the text, and we do not repeat them here. Rather, we focus here on long-term work that is necessary to make JaldiMAC complete and deployable.

Our innovative method of deployment greatly reduces loss caused by interference and so we anticipate low loss patterns as observed by Sheth et al. [38] for rural long distance links. In case the loss rate turns out to be non-negligible, we will integrate various link-layer recovery schemes such as ARQ and FEC in the next iteration of JaldiMAC.

Additionally, there is a wide range of physical parameters that leave ample room for tuning: varying channel width, adapting the bitrate, adjusting transmit power, and switching frequencies. It may be particularly useful to adjust these parameters dynamically.

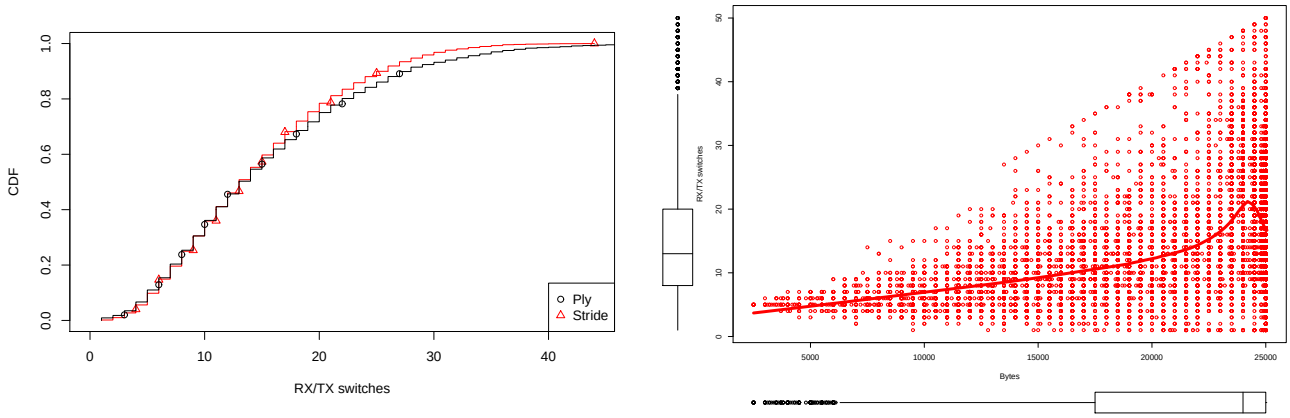
For our prototype, we plan to modify the open-source ATH9k driver [3] to enable flexible configuration of low-level parameters. In particular, we envision to prototype JaldiMAC as a Click [22] kernel module that offers the user a different virtual interface per service model or class, exposing a clear interface that enables fine-grained quality of service support.

We plan to deploy two testbeds in parallel – one by AirJaldi in rural India and one in the Bay Area, California, where we already have several 802.11n long distance links in place. By comparing the two testbeds we expect to distill insights about the variations in interference of urban versus rural environments. As soon as the prototype reaches maturity, AirJaldi will begin deployment for links throughout rural India.

9 Conclusion

Although WiFi is a cost effective and promising technology for extending broadband Internet connectivity to poor and sparsely populated communities, its proliferation remains slow. Our first contribution represents a study of the economic challenges faced by small rural WISPs. We identify towers as the biggest barrier to entry and suggest use of mountains or hills instead for placement of the base stations, even if the subscribers are far away. Our method of deployment is distinctive in its use of directional antennas to minimize interference while maximizing the service area.

In order to enable our proposed deployment topology, we contribute JaldiMAC, a new MAC architecture that achieves the following goals: *(i)* enable point-to-multipoint setups while harnessing the broadcast characteristics to its advantage, *(ii)* adapt to the asymmetry of



(a) RX/TX switches for ply and stride.

(b) RX/TX switches as a function of round size for ply.

Figure 9: Number of RX/TX switching per round.

Internet traffic, and *(iii)* provide loose quality of service guarantees for latency-sensitive traffic without compromising fairness.

Our simulator-based evaluation of JaldiMAC suggests that it fulfills its design goals. Its innovative scheduler is able to provide a 71% decrease in jitter and superior latency characteristics in exchange for a 5% increase in average RX/TX switches, as compared to a standard stride scheduler. Overall, we find that JaldiMAC performs surprisingly well at this early stage.

Acknowledgments

We thank Rabin Patra and Sergiu Nedevschi for the long brainstorming sessions and their guidance. The measurement data from their past experiments with long-distance links, along with the WiLDMAC source-code are invaluable, as are their comments while reviewing the draft for this paper. We also thank Anmol Sheth who joined our discussions over the phone and provided the draft for the yet unpublished paper[19]. Matt Podolsky and Matt Tierney for the details about ATH9K and directions for packet injection, although we run out of time before we could implement the protocol using real radios. Finally, we thank Teresa and Yael for their proof reading targeting earthly humans and not CS people.

References

[1] AirJaldi.Org - Wireless Network, Dharamsala, India. http://www.airjaldi.org.

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

[3] ath9k driver. `http://linuxwireless.org/en/users/Drivers/ath9k`.

[4] Freifunk Community Wireless Mesh Network, Berlin, Germany. http://start.freifunk.net/.

[5] Tarana Wireless, Berkeley, CA. `http://www.taranawireless.com/tech`.

[6] The AirJaldi Mesh Router, AirJaldi.Org, Dharamsala, India. http://drupal.airjaldi.com/node/9.

[7] The United Nations Population Database - World Urbanization Prospects: The 2007 Revision. http://esa.un.org/unup/.

[8] Ubiquiti Networks. `http://www.ubnt.com`.

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

[10] Ashutosh, Nirav, and Bhaskaran. Implementation and evaluation of a tdma mac for wifi-based rural mesh networks. page 6, 2009.

[11] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Trans. Netw.*, 5(6):756–769, 1997.

[12] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 2–11, New York, NY, USA, 1995. ACM.

[13] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a Media Access Protocol for Wireless LAN's. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 212–225, New York, NY, USA, 1994. ACM.

[14] E. Brewer, M. Demmer, B. Du, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, S. Surana, and K. Fall. The case for technology in developing regions. *IEEE Computer*, 38(6):25–38, 2005.

[15] P. Calyam and C. Lee. Characterizing voice and video traffic behavior over the Internet. In *International Symposium on Computer and Information Sciences (ISCIS)*, 2005.

[16] Cisco. Implementing QoS Solutions for H.323 Video Conferencing over IP. `http://www.cisco.com/en/US/tech/tk543/tk757/technologies_tech_note09186a0080094968.shtml`.

[17] Cisco. Voice Over IP - Per Call Bandwidth Consumption. `http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml`.

[18] R. Gummadi, R. K. Patra, S. Nedevschi, S. Surana, and E. A. Brewer. A radio multiplexing architecture for high

throughput point to multipoint wireless networks. In C. E. Perkins, E. M. Belding, and R. Jain, editors, *Wireless Networks and Systems for Developing Regions*, pages 47–52. ACM, 2008.

[19] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. 802.11 with Multiple Antennas for Dummies, 2010.

[20] IEEE 802.16. IEEE 802.16 Wireless MAN Standard for Wireless Metropolitan Area Networks.

[21] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. Technical Report TR-301, DEC Research, September 1984.

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

[23] R. Lenagala and Q.-A. Zeng. Study of dynamic mac layer parameters for starvation prevention in the ieee 802.11e mac layer protocol. In *Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006). IEEE International Conference on*, pages 124–131, June 2006.

[24] S. Marwaha, J. Indulska, and M. Portmann. Challenges and recent advances in qos provisioning, signaling, routing and mac protocols for manets. In *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*, pages 97–102, 2008.

[25] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, page 297. ACM, 2001.

[26] S. Nedevschi, R. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer. An adaptive, high performance mac for long-distance multihop wireless networks. In *ACM MOBICOM*, 2008.

[27] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. Softmac - flexible wireless research platform. In *Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[28] S. Park and D. Sy. Dynamic control slot scheduling algorithms for tdma based mobile ad hoc networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7, Nov. 2008.

[29] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks. In *NSDI*, 2007.

[30] R. Patra, S. Surana, S. Nedevschi, and E. Brewer. Optimal Scheduling and Power Control for TDMA based Point to Multipoint Wireless Networks. In *ACM SIGCOMM Workshop on Networked Systems for Developing Regions (NSDR 2008)*. ACM, 2008.

[31] K. Paul, A. Varghese, S. Iyer, and B. R. A. Kumar. WiFiRe: Rural Area Broadband Access Using the WiFi PHY and a Multisector TDD MAC. *New Directions in Networking Technologies in Emerging Economics, IEEE Communications Magazine*, 2006.

[32] 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.

[33] B. Raman and K. Chebrolu. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 156–169, New York, NY, USA, 2005. ACM Press.

[34] A. Rao and I. Stoica. An overlay mac layer for 802.11 networks. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148, New York, NY, USA, 2005. ACM Press.

[35] K. A. Rudre, P. S. Iyer, P. P. Kulkarni, K. A. Rudre, P. S. Iyer, and P. P. Kulkarni. MH-WiFiRe: Multi-Hop Extension to WiFiRe, 2009.

[36] A. Sharma and E. M. Belding. Freemac: framework for multi-channel mac development on 802.11 hardware. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 69–74, New York, NY, USA, 2008. ACM.

[37] A. Sharma, M. Tiwari, H. Zheng, and S. U. C. Barbara. Madmac: Building a reconfigurable radio testbed using commodity 802.11 hardware. In *First IEEE Workshop on Networking Technologies for Software Defined Radio (SDR) Networks*, September 2006.

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

[39] S. Surana, R. Patra, S. Nedevschi, and E. Brewer. Deploying a Rural Wireless Telemedicine System: Experiences in Sustainability. *Computer*, 41(6):48–56, 2008.

[40] S. Surana, R. Patra, S. Nedevschi, M. Ramos, L. Subramanian, Y. Ben-David, and E. Brewer. Beyond pilots: keeping rural wireless networks alive. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 119–132, Berkeley, CA, USA, 2008. USENIX Association.

[41] C. A. Waldspurger and W. E. Weihl. Stride Scheduling: Deterministic Proportional-Share Resource Management. Technical report, 1995.

[42] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *IEEE Communications Magazine*, 39(4):52–58, 2001.

A Appendix

```
def ply(layoutSize, classes):
  layout = createEmptyLayout(layoutSize)
  class = classes.pop()
  pos = 0, allocated = 0
  while pos < layoutSize and allocated < classes.allocation:
    layout[pos : pos + class.minChunk] = class
    pos = pos + class.minChunk + class.period
    allocated = allocated + class.minChunk
  if classes is not empty:
    nextLayout = ply(layoutSize − allocated, classes)
    nextLayoutPos = 0
    for pos in range(0, layoutSize)
      if layout[pos] is not empty:
        continue
      layout[pos] = nextLayout[nextLayoutPos++]
  return layout
```

Figure 10: Pseudocode of the ply algorithm in a Python-like language.