Challenges: Wide-Area wireless NETworks (WANETs)

Suresh Singh
Department of Computer Science
Portland State University
Portland, OR 97207
singh@cs.pdx.edu

ABSTRACT

A WANET is a wireless network where the wireless nodes can be located anywhere over the globe. However, the underlying design is such that the nodes believe they are part of a single-hop or multi-hop wireless network at the PHY and MAC layers. This is accomplished by using Software Defined Access Points (SoDA) that are based on the idea of Software Defined Radio (SDR). For the uplink, each SoDA samples the down-converted channel using an ADC (analog-to-digital converter). The sampled data is then multicast to the other SoDAs via the Internet. At each end-point, the received digital signals from the other SoDAs are summed and sent through the DAC (digital-to-analog converter) and transmitted on a designated channel after upconversion. In effect what we have done is to mix the RF environment at geographically separate locations (albeit with a time shift). This simple technique leads to a whole new model for designing and using wireless networks. Indeed, we can think of these networks as following the end-to-end model where only the end nodes understand the PHY/MAC layers and the access points and Internet serve simply as dumb relays. This paper describes a series of network models that are enabled and outlines the open research challenges.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication

General Terms

Design, Algorithm

Keywords

Software Radio, Access Point, Flexible PHY

1. INTRODUCTION

Imagine a wireless network where nodes are spread out across the globe but behave exactly as if they were part of a single-hop or multi-hop wireless network at the PHY and MAC layers. In Figure 1(a) for instance, nodes A, B, and C may be located anywhere but they form a 1-hop wireless network as shown in Figure 1(b). This is accomplished by using Software Defined Access Points (SoDA) built using Software Defined Radio (SDR) technology. For example, for the uplink, SoDA_A samples the wireless channel from node A and multicasts this data to SoDA_B and SoDA_C. For the downlink at SoDA_A, the digital streams received from the other two SoDAs are added, converted to analog, upconverted and transmitted on to node A. This technique mixes the RF environment at three geographically separate locations to create a single-hop wireless network as shown in Figure 1(b). Observe that the only nodes that understand the PHY and MAC layers used by nodes A, B, and C are these nodes themselves. The SoDAs are not involved in any data communication with these nodes – they simply sample a channel on the uplink and transmit on a given channel for the downlink and forward data over the Internet to other SoDAs. The methodology illustrated in Figure 1 may be extended to create multi-hop wireless topologies in a natural way. For example, if we wanted to create a linear topology $A - B - C$, the only change that we make is for the SoDA at A and C to exchange data only with the SoDA at B while the SoDA at B send's its sampled signal to both A and C's SoDAs.

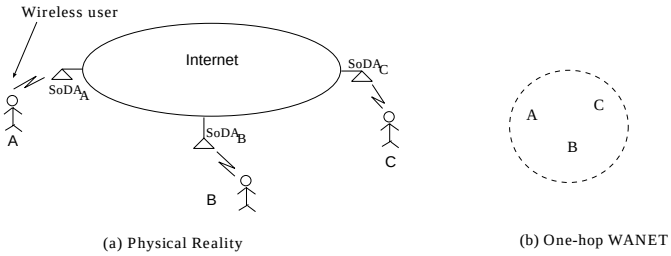


Figure 1: Example of a wide-area 3-node single-hop wireless network.

The innovation in this paper is that of building WANETs, based on the deployment of SoDAs. It is noteworthy that software-defined base stations are being increasingly used in the cellular community because of the capability to support multiple protocols and easy adaptivity to emerging standards. These same benefits apply to SoDAs as well. Indeed, given appropriate hardware (such as wideband ADC and fast computational engines), we can envision a single SoDA simultaneously supporting multiple IEEE datacom standards

MobiCom'08, September 8–12, 2008, San Francisco, USA.

(though that is not the focus of this paper). We argue that the WANET/SoDA architecture proposed here enables innovation because:

- By effectively circumventing the PHY/MAC functionality of the access points, we are enabling end-systems complete freedom in the form of PHY/MAC and other protocols they wish to run. Thus, we can envision the development of a variety of custom-tailored (open source or proprietary) PHY/MAC protocols for various end applications. Giving application developers control of the PHY and MAC layers (without being constrained by IEEE 802.x standards) may well lead to significant future innovation since we can now have true multi-layer optimizations.
- The FCC (Federal Communications Commission) is interested in opening additional frequency bands for unlicensed operation in addition to allowing co-existence of licensed and unlicensed users. However, they do not want to stifle innovation and are therefore soliciting new rules of use. The idea of using a generic infrastructure like SoDAs allows unlimited freedom in innovation and frees us from the inertia of standardization processes. This, we believe, is another big benefit of WANETs and SoDAs.

Another benefit, albeit a minor one, is that of providing opacity and privacy. For instance, interested groups of individuals can create their own WANETs which can be made relatively unintelligible to people outside the group by dynamically changing the PHY, and so on.

1.1 Paper Organization

The remainder of this paper studies the capabilities, limitations, and directions for WANET research. In the next section we first provide a proof-of-concept of a small WANET and explore some of the limitations. Section 3 then explores the issues surrounding the MAC design for single-hop WANETs. Multi-hop WANET design is described in section 4. We summarize the main ideas in section 6 after a brief literature review in section 5.

2. PROOF-OF-CONCEPT AND EXPERIMENTATION

The basic building block in a WANET is the the establishment of a communication channel between two distant wireless nodes via SoDAs (e.g., nodes A and B in Figure 1). Therefore, we need to determine if such a link can be established and the properties of such a link. By a "link" we mean specifically that data communication should occur between the nodes A and B. The experimental set up we use is shown in Figure 2 and it consists of two SoDAs and two nodes. Each SoDA consists of two radios[1] connected to a laptop. We use two radios since one is used for the uplink and the other for the downlink. Each of the two user devices A and B also consist of two radios connected to a laptop. The two SoDAs are connected to the departmental LAN using ethernet and communicate with each other using standard UDP/IP.

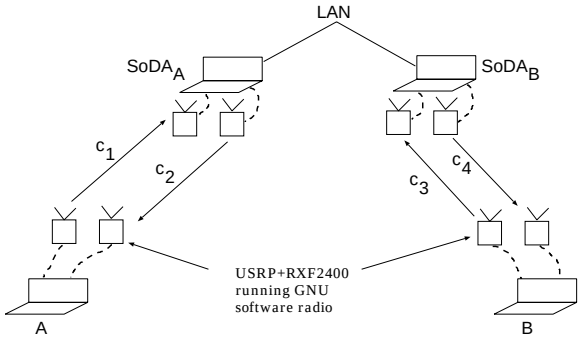


Figure 2: Experimental setup for verification of the "pipeline".

As shown in Figure 2, we use channel c_1 for uplink communication from A to SoDA$_A$ and c_2 for the downlink. Similarly, channels c_3 and c_4 are used between B and SoDA$_B$. On channel c_1, SoDA$_A$ receives RF signals, downcoverts to IF (Intermediate Frequency), digitizes the signal via an ADC, downconverts to baseband, and outputs the complex baseband data $x + iy$ where x is the I (Inphase) value and y is the Q (Quadrature) value. On the downlink channel c_2, the reverse process occurs with SoDA$_A$ receiving the I& Q signal from SoDA$_B$, upconverting to IF, convert to analog via DAC, upconvert to RF and then transmit. Nowhere in this process do the SoDAs attempt to detect or decode the channel symbols. Indeed, we modified the code for the SoDAs to ensure that they did not perform any carrier sensing either.

In the initial set of experiments, we simply send packets of various sizes from A to B. The radios attached to laptops A and B run the basic CSMA protocol while the radios that form the SoDAs use the software we modified to implement the *pipeline*. We noticed no problem in packet transfer across all sizes. We also add variable delay in the pipeline ranging from 5ms to 50ms to mimic Internet latency. Again, packet transfer is not affected. This shows that, at the very least, the pipeline works for 2 nodes.

The key idea illustrated in Figure 1(b) is that we can connect n nodes in a single-hop network using SoDAs by summing the I&Q data streams together prior to transmission on the downlink. Therefore, the next experiment that we conduct is to study the feasibility of forming a three-node single-hop wireless network. To this end, we add node C and SoDA$_C$ to the set up shown in Figure 2. Thus, node A's I& Q data collected on c_1 by SoDA$_A$ is multicast to SoDA$_B$ and SoDA$_C$. For downlink transmission to node A, SoDA$_A$ adds up the I& Q streams emanating from SoDA$_B$ and SoDA$_C$ and then transmits that to A on channel c_2. In adding up the two streams, SoDA$_A$ adds pairs of I values (one from each stream) to get a new I value and similarly for the Q values as well. No effort is made to synchronize timing of the streams.

In section 3.1 we discuss the impact of summing streams

[1]We use Universal Software Defined Radios (USRP) [1] running the GNU Software Radio toolkit. The USRP consists of a motherboard with four 12-bit 64Msamples/sec ADCs, four 14-bit 128Msamples/sec DACs, a million gate FPGA, and a programmable USB 2.0 controller. We use 2.4 GHz RF front-ends (RFX2400) implemented as daughter boards and attached to the motherboard. These radios run in the the 2.3 – 2.9 GHz frequency range. Programming these radios is relatively easy and requires a combination of C++ blocks of code (signal processing blocks) and Python (to tie the blocks together).

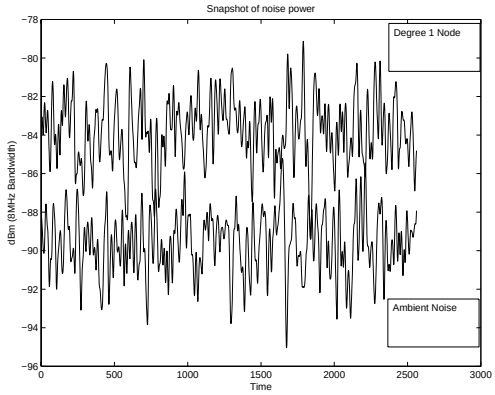


Figure 3: Illustration of additive noise.

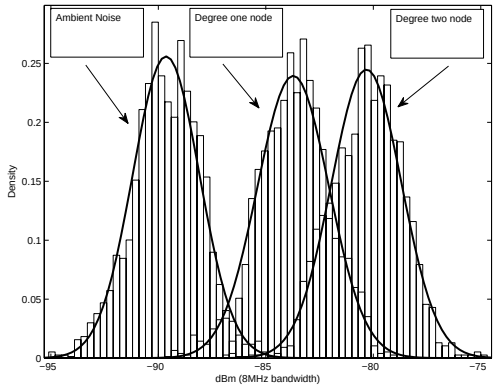


Figure 4: Noise distribution.

on total *noise* in a more formal manner. For our purposes here, it is instructive to look at Figure 3 and 4. In Figure 3 we see a snapshot captured by a Tektronix RSA 3308A Spectrum Analyzer. The lower plot shows the ambient noise level[2] and the upper plot shows the noise for the case of two nodes (i.e., each node has degree one) from Figure 2. We see that the noise at a node of degree one is higher than ambient noise. This is because of two factors:

1. The I& Q data that is sent on the downlink contains noise in addition to the desired signal (if a packet was being sent). The SoDAs perform no filtering because *they are not receiving data or decoding channel symbols*! Therefore, noise from one place gets recreated at another.

2. The second source of the additional noise is the SoDA hardware itself. The noise comes from each of the components (antenna, amplifier, mixer) and the ADC (quantization noise – since analog data is being represented as a certain number of bits, the lack of resolution manifests itself as noise). In our case, the SoDA noise is added on the uplink and then on the downlink (note that the DAC does not add to noise on the downlink).

[2]The mean is -89.63 dBm and the channel bandwidth is 8MHz giving us a value of approximately -160 dBm/Hz. This is close to the actual value of -174 dBm/Hz. The difference comes about because of noise in the Spectrum Analyzer hardware itself (the technical specifications from the Tektronix website detail the noise figure values).

We measure noise for the case of a degree two node as well, and as expected, the noise here is higher than in the degree one case. Figure 4 plots the pdf (probability density function) of ambient noise, noise when we have a degree one node and noise for a degree two node. The measured noise parameters are:

	μ	σ^2
Ambient Noise	-89.63 dBm	2.42
Degree 1 Node	-83.72 dBm	2.77
Degree 2 Node	-80.31 dBm	2.62

We note that the *noise for degree 2 is double that of degree 1*, as we would expect, since we are adding together two noisy streams. In general, if we add together n streams, the mean noise power will be n times higher. If we pass the reconstructed analog output stream through an amplifier just prior to transmission on the downlink, then the noise power will be higher still (based on the amplification factor). However, we do not do that in our current implementation.

Finally, we wanted to study the impact that noise has on the *bit error rate* of the user-level data. To do this, we reverted back to the experimental setup in Figure 2 and did the following. Node A sends a packet of length 1500 bytes that is received at node B. We ensure that node B does not perform any error correction. Node B takes the received packet and sends it back to node A, and this ping pong continues until the packet gets too garbled to be received. Figure 5 plots the average number of characters in error as a function of the number of times a packet is received. As is evident, as the packet is forwarded without error correction, errors accumulate linearly showing the cumulative effect of noise.

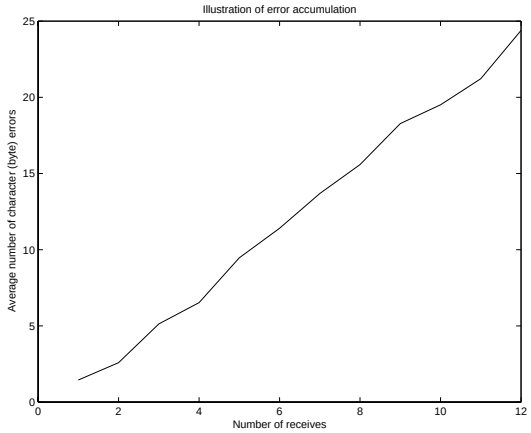


Figure 5: Noise for different network sizes.

2.1 Implementation Considerations

In general, a SoDA may be assisting in more than one WANET simultaneously. Assume that a SoDA is managing k wireless connections, all connected to one or more WANETs. For each connection, it assigns a pair of channels – one for the uplink and one for the downlink. Let us denote these pairs as $\cdots, < u_i, d_i >, \cdots$. The SoDA main-

tains a table that consists of pairs,

$$\{< u_i, d_i >, < (\text{ip}_1 : p_1), (\text{ip}_2 : p_2), \cdots, (\text{ip}_n : p_n) >\}$$

Data from u_i is *multicast* using standard IP Multicast to all the n IP addresses/port combinations. Similarly, data streams originating from each of the n IP addresses are *summed* and then transmitted on d_i. In order to sum up the data streams received from other SoDAs, the SoDA first retrieves the data within each IP packet. These I & Q values are saved in a FIFO buffer – one buffer per source IP. Values are removed from the buffer at the other end at a rate determined by the transmission rate over d_i and summed. These values are then transmitted on the downlink channel d_i. Note that even if no packet transmission is taking place, there is still a constant stream of I and Q values due to noise. However, sometimes, a buffer may be empty due to jitter or network congestion. This does not cause any problem except that the signal quality on d_i will improve (since there is less noise being transmitted). A key point to remember here is that everything happens in *real-time* with the obvious implications for the implementation. Also note that a node may well be participating in multiple WANETs simultaneously – it simply has a unique pair of u/d channels assigned for each WANET it is in.

2.2 Feasibility

The practicality of the ideas presented here depend heavily on what is possible in hardware today and in the future. The key elements of the hardware are the ADC/DAC and computational engines for running the PHY/MAC software. Our current implementation is based on the USRP boards which have 12-bit 64 Msamples/sec ADCs which give us a theoretical channel bandwidth of 32MHz (though it is less in practice due to filter rolloff). Unfortunately, however, in the current boards, all computation is pushed to an attached computer via a USB 2.0 connection. Since the maximum bandwidth of the USB is 32 MByte/sec, we can only use a channel of 8 MHz width.

Today, high-speed CMOS ADCs running up to 1.5 Gsamples/sec are available (if we use GaAs we can get up to 8 Gsamples/sec though with two orders magnitude increase in power draw) [3]. Thus, for our applications, the ADC technology is not the limitation. The real problem is the computational engine. Ideally, a SoDA would have embedded multi-core processors (due to the parallel nature of most computation involved) on board connected to the RF chain via fast interconnects. With this type of hardware, we can truly realize a full featured SoDA.

A second consideration is the load placed on the Internet by data exchange between SoDAs. In the current implementation, each SoDA generates 32MBytes/sec of data. If we build a WANET as in Figure 1 with n nodes, this puts a total load of $32n$ MBytes/sec on the Internet. In the future, as we get much higher wireless bandwidths, this load will only increase. However, there are a few approaches that will help tame this problem. If we only use SoDAs as a flexible access point, then all computation is done on-board so there is no load on the Internet. Indeed, PHY/MAC software can be loaded onto a SoDA in an on-demand basis depending on user needs. The load becomes an issue only when we consider WANETs. For this case, we will discuss two mechanisms in later sections that will reduce load – mixers (section 3.3) and multi-hop topologies (section 4).

3. SINGLE-HOP WANET

The discussion above shows that data volume and additive noise pose a fundamental constraint on the deployment of efficient WANETs. As we will see below, Internet latency/jitter also add complexity to the deployment of these types of networks. In this section, we discuss the impact of these constraints on MAC design and propose possible solution approaches.

Before proceeding with the discussion, we note that if multiple WANET nodes are present within the service area of a common SoDA, they can either all compete for a common uplink and downlink channel or they can each be assigned a unique u/d channel pair. In this paper we consider the latter case only for ease of exposition. The case when multiple nodes compete for the uplink/downlink at each SoDA can be studied in a similar fashion.

3.1 The Problem with Additive Noise

Assume that a single-hop network has $k+1$ nodes and we are looking at the signal received by node $k+1$. Initially *assume that there are no ongoing data transmissions*. Regardless of this, the SoDA's continuously sample the channel and transmit the received signals. In this case, the signals being sampled will all be noise. Denote the signals sent by $\text{SoDA}_1, \cdots, \text{SoDA}_k$ to SoDA_{k+1}, via the Internet, as $n_1(t-\tau_1), \cdots, n_k(t-\tau_k)$ (note that each SoDA multicasts its signal to all other SoDAs, but here we look only at SoDA_{k+1}). τ_i denotes the one-way latency from node i to node $k+1$. The term n_i denotes the sum of the environmental noise received at SoDA_i *plus* the noise in the RF chain at SoDA_i that includes noise in the antenna, amplifier, downconverter, and ADC (quantization noise). At SoDA_{k+1}, these signals are summed and transmitted. A simple model for the signal received by wireless node $k+1$ is then,

$$n_{k+1}(t) + G_{\text{SoDA}} G_R \alpha\gamma \left(\sum_{i=1}^{k} n_i(t-\tau_i) + \bar{n}_{k+1} \right)$$

where, α is the amplification prior to transmission (in case there is any) by SoDA_{k+1}, γ is the pathloss to the receiver, G denotes the antenna gains, and $\bar{n}$ denotes the noise in the RF chain at SoDA_{k+1}. The term $n_{k+1}(t)$ is the environmental noise at the receiver plus the noise in the receiver's RF chain. G_{SoDA} and G_R respectively denote the antenna gains at SoDA_{k+1} and wireless node $k+1$.

Let us now assume that node k is transmitting a signal $s(t-\tau_k)$. It is received by SoDA_k (after attenuation) as $\bar{s}(t-\tau_k) + n_k(t-\tau_k)$. At node $k+1$, the received signal is then,

$$n_{k+1}(t) + G_{\text{SoDA}} G_R \alpha\gamma \left(\sum_{i=1}^{k} n_i(t-\tau_i) + \bar{s}(t-\tau_k) + \bar{n}_{k+1} \right)$$

We can write the SNR (Signal to Noise Ratio) at node $k+1$ as,

$$\begin{aligned} SNR &= \frac{\left| G_{\text{SoDA}} G_R \alpha\gamma \bar{s}(t-\tau_k) \right|}{\left| n_{k+1}(t) + G_{\text{SoDA}} G_R \alpha\gamma \left(\sum_{i=1}^{k} n_i(t-\tau_i) + \bar{n}_{k+1} \right) \right|} \\ &\leq \frac{|\bar{s}(t-\tau_k)|}{\left| \bar{n}_{k+1} + \sum_{i=1}^{k} n_i(t-\tau_i) \right|} \end{aligned} \tag{1}$$

As an illustration, if $k = 2$ then the extra noise reduces

SNR by 3dB (see Figure 4). If $k = 100$, then the SNR is down by 20dB which affects the achievable data rate for a given BER. If we apply Shannon's capacity theorem to these values, we see that maximum achievable capacity reduces by $W \log_2 k$. For $k = 100$ this is a reduction of $6.6W$ bps where the bandwidth of the channel is W Hz. This is a significant drop in throughput due to additive noise and hence the size of single-hop WANETs cannot be too large.

3.2 The Problem with Latency and Jitter

Let us assume that we have set up a one-hop WANET with n nodes. The problem of then having an efficient MAC protocol to enable communication is interesting for the following reasons:

- *Large Latency:* Recall that SoDA's use UDP/IP to multicast packets containing the I&Qdata to all participating SoDA's. Therefore, the delay experienced in the Internet can be thought of as the propagation delay τ between nodes. Unlike most single-hop wireless networks, the τ value in WANETs is very large (of the order of tens of milliseconds for nodes spread out across the country).

  In order to see the impact of the large values of τ on throughput, consider the throughput of a simple protocol like NP-CSMA (Non-Persistent CSMA). Let us assume that τ is the maximum value of one way delay among all pairs of nodes. Assume that packet length is $T \geq \tau$. The throughput of NP-CSMA [5] under poisson traffic assumptions can be written as,

$$S = \frac{gTe^{-g\tau}}{g(T + 2\tau) + e^{-g\tau}}$$

  where g is average load in pkts/sec. Since latency in the Internet can be of the order of tens of milliseconds, let us assume that $T = \tau$ (at a transmission speed of 11Mbps, this translates to 13KByte packets for $\tau = 10ms$). Then, the throughput is *less than* $\frac{e^{-gT}}{3}$ – a very low value. Indeed, if $\tau > T$, the throughput is smaller still. Two points stand out from the above example – CSMA type protocols may not be appropriate for WANETs and we will most likely need explicit ACKs.

- *Range of Latencies:* Consider a three-node network with nodes A, B, and C located in the Internet such that $\tau_{BC} \geq 2\tau_{AB} = 2\tau_{AC}$. In other words, the delay between B and C is at least twice as long as the delay between AB and AC. Consider a contention-free token-ring MAC protocol where a node transmits and then it is the next node's turn to transmit. Say node C transmits at *time 0* and it is next A's turn. C's transmission arrives at A at time τ_{AC} after which A transmits. A's transmission arrives at B at time $\tau_{AC} + \tau_{AB} = 2\tau_{AC}$. It is now very likely that C's transmission will arrive at B at a time overlapping A's transmission (since $\tau_{BC} \geq 2\tau_{AC}$) and cause a collision thus violating the contention-free property.

  There is a fix for this problem but it is very inefficient. Let τ_{max} be the maximum value for pairwise delays. Then, in the above example, C transmits at *time 0*. A receives the transmission at time τ_{AC} but does not begin transmission until time τ_{max} (in other words, after receiving the packet from C, A waits $\tau_{max} - \tau_{AC}$ before transmitting). A's transmission arrives at B at time $\tau_{max} + \tau_{AB}$ and B transmits at time $2\tau_{max}$. Note that there is no packet collision at B.

  Clearly, this protocol can be made to work for arbitrary n but it will be very inefficient because of the need to synchronize all transmissions to τ_{max}. Furthermore, this protocol relies on knowledge of τ_{ij} and for this value to be stable – neither of which is a reasonable assumption.

- *Jitter:* Different packets between the same pair of nodes may experience different amounts of delay τ. This is unlike typical wireless networks where there is no jitter between pairs of nodes (unless they move, in which case this value changes slowly relative to speed of signal propagation).

  If we return to the CSMA discussion from above, the problem with jitter becomes clear – the value of T will need to be based on the *worst case* latency τ between any pair of nodes. Thus, if there is a great deal of jitter, then the largest value of delay will need to be used for packet size T. This again underscores the need to develop protocols that are not based on CSMA.

- *Loss:* Packets may well be lost in transit through the Internet which will make the transmissions at the other end nonsensical. Furthermore, packets sent from node A may reach B but some may not reach C. This means that the "system state" seen at different nodes of the WANET may be somewhat different. Again, we generally do not have this problem in single-hop networks where the system state as seen by all nodes is consistent.

3.3 Discussion

Designing efficient MAC protocols for single-hop WANETs is an interesting problem, and in this section we briefly discuss some possible approaches that may be fruitful.

1. The variability in latency and inherent jitter cause problems for sophisticated protocols like CSMA. We believe that this variability can be exploited by using *simple* protocols such as ALOHA. The primary drawback of a protocol like ALOHA is the problems caused by overlapping transmissions. In our case, however, given the long latencies between nodes, it is likely that collision probabilities will be much smaller since packet sizes are a fraction of the latencies. However, it is important that every packet be ACK'ed as well because a sender has no other way of knowing if the transmission was successful.

2. We can make the following *key observation* about the system: the uplink transmissions from a user to its SoDA are interference-free (unless there are multiple users of the WANET located in the coverage area of the same SoDA sharing the uplink/downlink channels) since uplink and downlink transmissions occur on different channels. The collisions only occur when a SoDA sums up the signals from all other SoDAs prior to downlink transmission.

   Let us define a *mixer* as a piece of software, located somewhere in the Internet (possibly at an end-host),

that works in conjunction with the WANET. On the input end, the mixer receives data from n SoDAs (these are data streams corresponding to each uplink channel and therefore contain individual node transmissions) and on the output, it unicasts n data streams back to these SoDAs. The IP implementation of this mechanism is trivial since each SoDA and the mixer have a unique IP address. However, the interesting aspects of the implementation are in how the n data streams get combined to produce n output streams.

- *Blind:* This is the design we used in the proof-of-concept implementation. I and Q data streams arrive from SoDA's $1, 2, \cdots, n$ and each data stream is put into a FIFO buffer. n data streams are created from each of the SoDAs by summing values from each of $n-1$ buffers. Thus, for SoDA i, a data stream is created by summing values from all buffers except i (i.e., the data stream that originated from SoDA i itself). Thus, what this mixer does is simply move the task of combining streams away from SoDAs.
- *Threshold-Based:* Since the mixer has the I & Q values for each stream, it can combine the streams together more intelligently to reduce the possibility of collisions. One simple implementation of this idea can be to use power as an indicator of the start of a data packet in a stream. For a given stream, the mixer calculate the power value represented by each pair of I & Q values. If the power is greater than some threshold ρ for at least τ seconds, then the mixer decides that this is a packet transmission as opposed to noise. By performing this test for each of the n data streams as the data *arrives* into the buffers, we can put markers in each buffer indicating the deduced start and end of packets. The output streams are then created by carefully mixing the I& Q streams to minimize collisions. In order for this implementation to work, each buffer can be thought of as having a size B plus overflow size S. The overflow part determines how much a stream can be delayed in order to accomplish this form of intelligent mixing.

  It is important to keep in mind that this strategy does not guarantee collision free mixing. When there is high load or there are overlapping bursts of data transmissions, there will be collisions since the mixer cannot arbitrarily delay packets in the buffers while awaiting a lessening of load. Indeed, since the buffers are of fixed sizes $(B+S)$, there is an automatic constraint on the degree of flexibility possible.
- *Packet-Based:* Using power values to estimate the start and end of packet transmissions is not an entirely reliable strategy to determine packet boundaries since the power level depends on the received power at some SoDA receiver which, in turn, is affected by fading and transmit power. We propose building a more sophisticated mixer that decodes and receives each stream and then interleaves the packets from different streams to create n outgoing streams. The outgoing streams are again I& Q streams created by the mixer. Note that this does not get rid of collision problems since we are still operating in real-time and are thus constrained on how well we can interleave packets without collisions. However, we expect to reduce collisions as well as improve SNR for each stream produced.

In summary, it is easy to see that the set of challenges in building WANETs is significant and exciting. The unique benefits and challenges of the WANET architecture may lead to significant innovations in the mature field of MAC design.

4. MULTI-HOP WANETS

The discussion in the previous section raises an important concern – that of scalability of WANETs. As the number of nodes increases, the complexity of mixers increases at a faster rate. The storage complexity for buffering streams as well as the computational complexity for packing packets to create streams increases. Furthermore, as the number of nodes increases, the total noise also increases linearly as does network load.

In order to counter this increase in complexity, it makes sense to split the single-hop WANET and create a multi-hop WANET. However, the manner in which this is done can have a significant impact on complexity and performance. We note, however, that *unlike traditional MANETs whose topology is to a large extent determined by location, multi-hop WANETs can have any topology we choose.* This flexibility is a powerful tool in enabling scalability without sacrificing performance.

4.1 Architecture 1: Set of Single-hop WANETs

The most obvious way to reduce the complexity of a large WANET is to split it into several single-hop WANETs (i.e., cliques) and then provide some form of connectivity between them. Let us say that we have n nodes that are split into k single-hop WANETs of sizes $n_1, n_2, \ldots, n_k$. Then, we can connect these k WANETs together via a mixer as shown in Figure 6. Here, each of the k WANETs has their own intra-WANET mixer and there is one inter-WANET mixer. The inter-WANET mixer forwards incoming streams to all other intra-WANET mixers. In terms of implementation, we have $3 \times 3 = 9$ possible combinations depending on which type of mixer (Blind, Threshold-Based, or Packet-Based) is used for the inter and for the intra-WANET mixer implementations. At the least complex we can use blind mixers only, but this implementation gives us no benefit over connecting all n nodes in a single large WANET. Alternatively, using packet-level mixers everywhere gives us the highest throughput but at the cost of computational complexity. If the inter-WANET mixer is either blind or threshold-based, we can view it as a Layer 2 repeater in traditional LAN architectures. If it is a packet-level mixer, then we can view it is either a router or an intelligent repeater.

Consider an alternative approach to connecting together WANETs without the aid of any mixers whatsoever. The idea is to split the task of the mixer and delegate it to some subset of nodes in each WANET. Let us select one node from each of the k WANETs to serve this role of repeater. Then, one interesting architecture that comes to mind is to form an additional single-hop WANET consisting of only

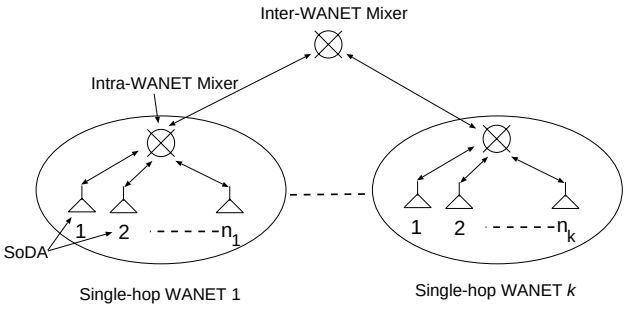


Figure 6: Using mixers to connect single-hop WANETs.

these selected repeater nodes. Each of these nodes will be assigned *two* upstream (u_1, u_2) and two downstream (d_1, d_2) channels to its SoDA. One pair of upstream/downstream channels u_1/d_1 is used to communicate with nodes in its own WANET while the other pair is used to communicate with the other $k-1$ repeater nodes in the other WANETs. The resultant architecture is shown in Figure 7. The benefit of this architecture is that it does not require any support from nodes in the Internet (e.g., mixers). In operation, a repeater node will forward packets (using u_2) on to the other repeater nodes only if the destination is not within its own WANET. The downlink d_2 at each of these repeater nodes consists of data streams generated by each of the other $k-1$ repeater nodes. A problem that may arise here is that two packets from different WANETs destined to other WANETs may well collide in this downlink at some or all repeater nodes. Therefore, we need to run an appropriate MAC protocol between the repeater nodes as well.

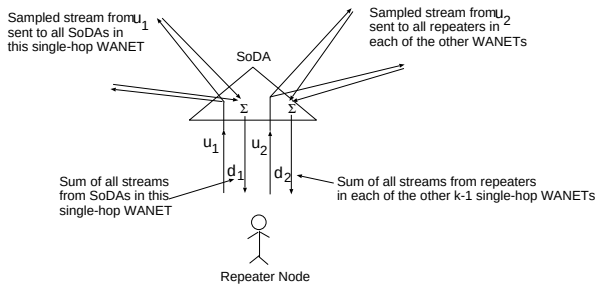


Figure 7: Using repeaters to connect single-hop WANETs.

We can generalize the connectivity of the k repeater nodes into any arbitrary graph. Consider one example shown in Figure 8 where the k repeater nodes are connected together in a ring. Consider node repeater nodes $i-1, i, i+1$ with the connectivity as shown in the figure. On the downlink d_2 at node i, the signals received from $i-1$ and i's SoDAs are summed. On the uplink u_2, any packet received on d_2 is retransmitted and forwarded to each of the neighbors $i-1$ and i. If the packet had initially originated at node i then that packet is not forwarded any more. The benefit of this topology is the reduction in packet collisions emanating from different WANETs. The drawback is the added delay in forwarding the packet to all $k-1$ WANETs. Clearly, other topologies are also possible that strike a balance between the probability of packet collision and forwarding delay.

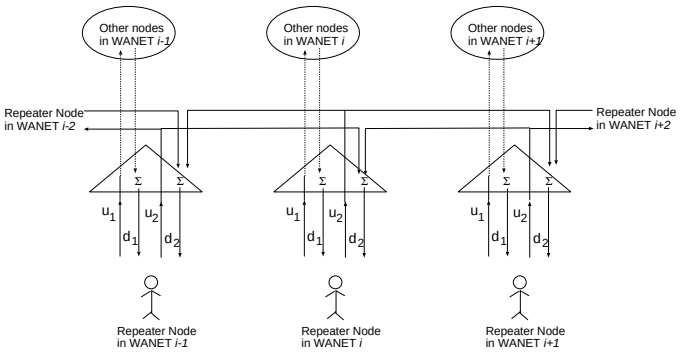
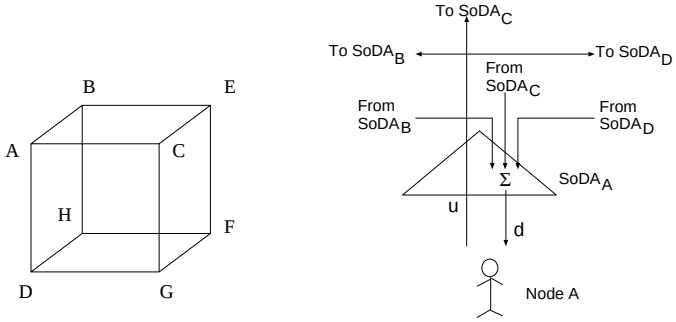


Figure 8: Using repeaters to connect single-hop WANETs in a ring.

4.2 Architecture 2: Arbitrary Topologies

We can generalize the idea illustrated in Figure 8 by considering arbitrary topologies where *each and every* WANET node is a repeater. For instance, consider the network topology shown in Figure 9(a) where each node has degree 3. The implementation of this topology for some node A is illustrated in Figure 9(b). The downlink to node A from SoDA_A consists of the sum of the data streams from each of the neighbors B, C, D. The upstream from A to SoDA_A is multicast to B, C, and D. Say node B has a packet for node D. Node B transmits the packet on its uplink and the data stream created by SoDA_B is multicast to B's neighbors A, E, H. At SoDA_A, the data streams from SoDA_B, SoDA_C, and SoDA_D are summed and transmitted on the downlink to node A. If there is no collision and A is able to retrieve B's packet, it transmits it on its uplink whereupon SoDA_A creates a data stream and multicasts it to SoDA_B, SoDA_C,, and SoDA_D. Node B does not forward this packet anymore (since it was the originator). Node D receives the packet (assuming no collisions from other data streams) and also does not forward it since it is the destination. In a naive implementation node C will forward the packet by multicasting it to E, A, G. However, this can be easily prevented by adding a TTL (Time To Live) field in the form of a hop count into the packet. Thus node C will drop the packet if the TTL is set to 2.



(a) 8-node topology (b) Implementation at A

Figure 9: 8-node network topology.

At this point, we need to make an important observation

regarding this form of forwarding. As noted previously, SoDAs are oblivious to the contents of data streams. Thus, we statically associate an uplink channel with an IP multicast address at each SoDA. In other words, in the example above, node A cannot selectively forward B's packet only to node D. All data on A's uplink is multicast to B, C, and D, regardless. If we wanted to give the nodes in the example the ability to *route* packets then the only way to do that would be to allocate three uplink channels from A to SoDA_A and associate each uplink channel with the appropriate IP address of B, C, and D's SoDAs. However, this is an expensive approach whose cost grows with the degree of a node.

We can now consider combining the architectures described in this section with those described in the previous section. For instance, in Figure 8 we connect single-hop WANETs in a ring topology. We could just as well connect single-hop WANETs in arbitrary multihop topologies such as in Figure 9 where nodes of the graph correspond to repeater nodes in each WANET. The benefit of doing this is to reduce delay seen in the ring network model.

4.3 Discussion

It is interesting to note that there are two conflicting goals in designing multi-hop WANETs. First, for a given number of nodes n, *minimizing delay* between source and destination of a packet requires a low-diameter topology. However, minimizing diameter leads to an increase in nodal degree (the extreme case is a fully-connected topology or a single-hop WANET). As node degree grows, the problem we face is that of increasing noise on the downlink and the potential for more packet collisions when streams get summed prior to transmission on the downlink. The problem can therefore be stated as follows: given some number of nodes n, design a topology that minimizes degree and diameter simultaneously. This problem is a restatement of the famous Moore bound [4] and has been well-studied in the Graph Theory literature. It is certainly interesting to consider the application of these results to WANET design taking into consideration practical limitations.

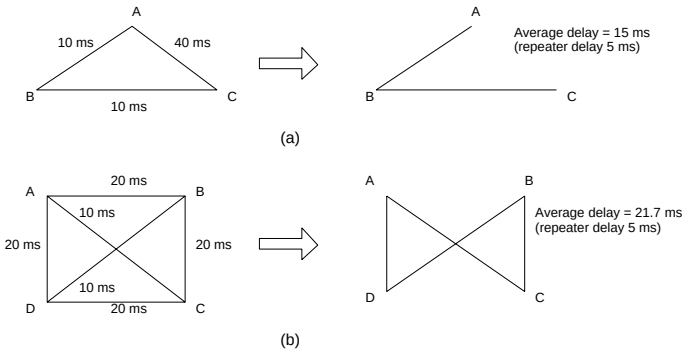


Figure 10: Effect of Internet latency on topology design.

Another aspect of topology design stems from variable Internet latencies between pairs of nodes. To see the impact of this, consider the two sample fully-connected topologies shown on the left in Figure 10. The numbers shown on the edges are the Internet latencies between the pairs of nodes. Let us further assume that the forwarding delay at a repeater is 5 ms. In Figure 10(a), if we use the fully connected topology as seen, the average delay is: $(10+10+40)/3 = 20$ ms. However, if we eliminate the link A-C, the average delay is $(10 + 10 + 25)/3 = 15$ ms. A slightly different example is shown in Figure 10(b). A ring topology reduces node degree but there are more than one choice. If we select the ring A-B-C-D-A then the average delay will be $(20 + 20 + 20 + 20 + 45 + 45)/6 = 28.3$ ms. On the other hand, for the topology shown, the average delay is 21.66 ms.

Finally, observe that we are in no way constrained to use undirected topologies only. There are some benefits to be gained by using directed graphs since the *in-degree* is the only parameter that affects noise on a downlink. Furthermore, Internet latencies may be non-symmetric which leads naturally to directed topologies.

5. RELATED WORK

Work that is closest to our SoDA model comes from the software basestation implementations (Vanu Inc.). For instance, [2] describes using ethernet as an interconnect between RF front ends and processing hardware.

6. CONCLUSIONS

In this paper we have proposed a new way of constructing wireless networks that can span the globe. The key idea is to use software defined access points to sample the RF environment and multicast it to other similar access points who forward it onward to the wireless node(s). We tested this basic idea in a prototype and characterized its behavior. This fundamental building block can result in a myriad of interesting architectures and protocol design problems, many of which are discussed in this paper.

Acknowledgements

I would like to thank our shepherd Victor Bahl for insightful comments that helped improve the paper. Many thanks to Meenaktchi Venkatachalam for her valiant efforts in getting the SDR software to run and to Philip Wong for providing the measurement set up. Part of this work was funded by the NSF under award CNS–0722008.

7. REFERENCES

[1] Universal software radio peripheral (usrp), http://www.ettus.com/.

[2] G. Britton, B. Kubert, and J. Chapin. Rf over ethernet for wireless infrastructure. In *SDR Technical Conference, Orange County, CA*, 2005.

[3] Kent H. Lundberg. High-speed analog-to-digital converter survey, http://web.mit.edu/klund/www/papers/. October 2002.

[4] M. Miller and J. Siran. Moore graphs and beyond: A survey of the degree/diameter problem. *The Electronic Journal of Combinatorics*, 2005.

[5] Raphael Rom and Moshe Sidi. *Multiple Access Protocols: Performance and analysis*. Springer-Verlag, 1989.