Wireless Communication Using Sound

CSE 561 Course Project
Phases 1-3

Group B
John P. John
Tanya Bragin
Tomas Isdal

December 12, 2005

Table of Contents

I. Introduction

The goal of this project is to build a wireless ad hoc network using sound waves as the means for communication. The signals are going to be sent with a microphone and received with a speaker.

II. Requirements

The system requirements are divided into three sets of increasing complexity. These requirements will be addressed in three corresponding phases.

Phase 1: Design and implement *reliable communication* between two PCs using a microphone and speaker.

Phase 2: Design and implement *robust ad hoc routing* between a set of PCs using sound as their only means of communication.

Phase 3: Design and implement *efficient and fair resource control* among a set of PCs using sound as their only means of communication.

III. Key Design Concepts Outline

Reliable Communication

Note: See Phase 1 report for detailed justification for these design choices.

1. **Basic Communication**
   - Signal encoding using frequency modulation at sender
   - Signal decoding PCM analog-to-digital conversion at receiver
   - Amplitude to frequency conversion using Fourier transform
   - Using clock signal to synchronize between sender and receiver (no clock drift)
   - Bit encoding using frequency pairs for 0 and 1 bits
   - Bit decoding using relative power difference on 0 and 1 frequencies

2. **Handling Transmission Errors**
   - 8-bit Cyclic Redundancy Check (CRC) to ensure packet integrity at receiver
   - Positive Acknowledgements (ACKs) to notify sender about successful transmission
   - Stop-and-Wait retransmissions at sender to ensure reliable transmission

Routing

Note: See Phase 2 report for detailed justification for these design choices

1. **Addressing**
   - Static 4-bit network addresses
   - No node discovery

2. **Ad Hoc Route Discovery**
   - On-demand route discovery using broadcast flooding
   - Route caching to minimize control information
   - Process to recover from route failures using Route Error packets

3. **Forwarding**
   - Local Route Table forwarding vs Source Route forwarding
     - Minimum Packet Size
     - Handling Route Loops

Resource control

1. **Sharing Air**
   - Carrier sense at the sender to detect channel availability
   - Exponential randomized back-off
   - RTS/CTS scheme was not chosen because it is only useful when data packet size is much larger than the RTS/CTS overhead (see page 10, “Sharing Air” for informal proof).
   - Because the audible sound channel is inherently lossy, transmitting very long packets is not practical, because of retransmission overhead, so RTS/CTS is not required.
   - Instead, we ensure max packet size using fragmentation and reassembly at the application.

2. **Attempt to hold down the unused air time and overhead traffic**
   - Fast transmission rate achieved by efficient signal processing and use of multiple data transfer channels
   - No regular routing updates
   - Carrier sense, no back-off if free
   - No ACK if collision occurred at the receiver
   - Random exponential back-off

3. **Nodes wanting to send get to do so in a reasonable time**
    - Fair backoff implemented by pausing the retransmit timer instead of restarting it, so nodes that have been waiting longer have shorter retransmit counter times

<u>Miscellaneous</u>

1. **Tricks to reduce the need to retransmit**
    - Most common reason for retransmission is for one bit in a couple of bytes per packet does not get recognized as either a 1 or a 0
    - Forward error correction using a parity bit per byte fixes this problem and dramatically reduces retransmission times

2. **High quality recovery from various errors and failures**
    - Recognizing and discarding packets at MAC layer

3. **Survivability for "normal" changing conditions (intermittent obstructions, moving nodes, routing path changes)**
    - Stop-and-Wait retransmissions to recover from intermittent obstructions and noise
    - Recover from node failure using Route Error packets and route re-discovery

4. **Survivability for abnormal changes ("jamming" by noise sources in "your" frequencies)**
    - Mechanism to dynamically vary frequencies used for data transfer to recover from persistent noise in a particular channel

5. **Reducing overhead traffic by (e.g.) ranking choices by probability, or maintaining info on the current state of the network.**
    - Route caching to minimize control information

6. **High performance (high bit rate, low errors)**
    - Avg. Bit Rate = 128bits/sec
    - Avg. Error Rate = 10%

7. **Overcoming sound-specific limitations (supporting distance up to 6 feet between microphone and speakers)**
    - Summing up FFT samples to cancel out white noise over distance
    - High-Frequency bands to avoid low-frequency constant noise sources (CPU fans)

8. **Internetworking with Mike’s Team’s network**
    - We built a gateway to tunnel packets from one network to another
    - Connects two heterogeneous networks
        - ▪ Our network (N1): On-Demand Source-like Routing
        - ▪ Connected network (N2): Distance-Vector Routing
    - Application-level data tunneling, since there is no common network or transport interface
    - Transparent to nodes on both networks
        - ▪ Requests N1 → N2: Gateway responds to route requests to any address in N2
        - ▪ Requests N2 → N1: Gateway periodically broadcasts itself as a node that has a path to any address in N1

9. **Usability**
    - To improve usability of our test application we implemented a GUI
    - Application/Network/MAC/Physical level message history
    - Destination/Message/Send Window for message entry
    - Automatically created network graph

IV. Phase 1 Design

Our design process focused on building the system from the ground up, starting with reliable communication and addressing, efficiency, ad hoc routing and resource control in later phases. Our design philosophy was to explore simple models first and through a method of iterative testing and re-design to arrive at a final model. Below we discuss main design decisions we had to make during Phases 1 of this project. We also anticipate alternative design decisions and additional optimizations.

<u>Basic Communication</u>

Basic communication was accomplished using the following principles:

- Frequency Modulation
- Pair-Frequency Bit Encoding
- Multiple Channels
- PCM Analog-to-Digital Conversion
- Fourier Transform Amplitude-Frequency Conversion
- Relative Power Bit Decoding
- Clock Signal Synchronization

We now explore these design principles in detail.

Representing Bits

<u>Signaling</u>
Signal is generated using a microphone attached to a sound card. We chose frequency modulation as the basic mechanism to accomplish signaling in our protocol. Amplitude modulation was discarded because we are operating in a wireless environment and the amplitude of the signal depends on the distance of the receiver from the sender and this distance is not guaranteed to be fixed. On the contrary, frequency of the signal does not vary with distance and can be recognized using Fourier Transform method in spite of noise.

<u>Encoding Scheme</u>
We chose to use a Pair-Frequency Bit encoding scheme to represent bits with signals. This encoding scheme involves using presence of Frequency#1 to represent bit 0 and Frequency#2 to represent bit 1. An alternative was to use a single frequency and represent bit 0 with its absence and bit 1 with its presence. We decided to discard this approach, because it seemed more susceptible to noise. A completely different approach is to encode a signal using transitions. We have not explored this option, because we were gave preference to simple design options first.

<u>Channel Utilization</u>
Clearly there are more than two frequencies available in the audible sound waves spectrum. In order to take advantage of the full spectrum, we decided to design our system such that it can send signals on multiple channels at once. For example, if there are 2 channels available for data transfer, we can send 8 bits at the same time. In practice we implemented 8 concurrent channels, however we also programmed the sender to dynamically fall back to using 4, 2 and 1 channel if needed (see Noise section).

Detecting Bits

<u>Medium Sampling</u>
Medium sampling is performed continuously using a speaker attached to a sound card. Pulse Code Modulation (PCM) sampling technique used by the sound card digitizes analog signals and provides the results in an amplitude wave form. Fast Fourier Transform (FFT) performed on this data outputs the power for each frequency at a point in time. This metric can be used to decide whether a particular frequency was heard and thus to decode the actual signals that represent the bits.

Signal Decoding
To decide whether we see a 0 or a 1 on a particular channel, the receiver compares the relative power of the frequencies representing each bit. If the frequency representing bit 0 is above the noise threshold and the frequency representing bit 1 is not, the receiver decides that it detected a 0 bit (and vice versa for bit 1). If both frequencies are below the noise threshold, the receiver decides that there is no signal. If both frequencies are above the noise threshold, the receiver compares relative strengths of the signals and if the difference is significant, chooses the more powerful one. This method has worked fairly well in decoding the bits in absence of noise. In addition, we made a couple of optimizations to account for distance and noise (see Noise section).

Synchronization

Clock Signal
In order for the receiver to know that two subsequent bits that are the same were received, there needs to be a synchronization mechanism. System clock cannot be relied upon for synchronization because of sound card sampling inaccuracy and process pre-emption. This phenomenon is referred to as clock drift. We decided to deal with this issue by adding a clock signal. The sender ensures that this signal alternates between a 0 and a 1 every time a bit is transmitted. The receiver looks for transitions to determine when to read the next bit from each of the data channels. An alternative is to encode transitions into the signal itself, such as Manchester encoding. However, based on our minimalist design philosophy, we pursued a more straightforward approach.

Transition Detection
Transitions are detected using a FFT sampling interval that is at least 4 times less than the length of the signal. If the receiver detects a 0 on the clock signal, the clock is set to 0. If the receiver detects no change in the signal or no signal at all, it assumes there is no transition. However, if the receiver detects a 1, there is a transition. This approach has the danger of “losing ticks” when the receiver fails to detect a clock signal due to noise or inefficiencies in signal decoding. We implemented an optimization to prevent the “losing ticks” phenomenon (see Noise section).

Reliable Communication

Reliable communication was accomplished using the following principles:

- Cyclic Redundancy Check
- Positive Acknowledgements
- Stop-and-Wait Retransmissions

Integrity Checks

Packet Level Check
Not all link-level signal loss can be prevented, so we built integrity checks into our packet structure. In addition to source and destination addresses and the data payload fields, the packet will also include an 8-bit Cyclic Redundancy Check (CRC) field that provides a mechanism for determining whether the data got corrupted during transit. This field is calculated at the sender and re-calculated at the receiver after packet reassembly.

Acknowledgements/Retransmissions

MAC Layer Reliability
We employ a Stop-and-Wait approach to ensuring reliability at the MAC layer. It is a common practice to provide this level of reliability at the MAC layer in wireless environments because of the inherent unreliability of the channel. The sender sends only one packet at a time and waits for an acknowledgement from the receiver before sending the next. The receiver only sends the acknowledgement if the CRC check passed. The sender employs a timeout mechanism to re-send an outstanding packet if it did not receive an acknowledgement. We designed a 1-bit sequence number in the packet so that the receiver can distinguish when a packet is a re-transmission. Since there is only one outstanding packet at a time, a 1-bit sequence number is sufficient.

V. Phase 2 Design

Ad Hoc Routing and Robust Communication was accomplished using the following principles:

- Summing FFT Samples
- High-Frequency Bands
- Minimize transmission of routing control information
- Minimize network packet size

We now explore these design principles in detail.

Robust Communication

In phase 1 we implemented a stop and wait protocol to ensure reliable transmission. However, in order to communicate effectively, hosts need to receive signals with a reasonably high probability of success. In our project we are assuming that hosts need to be able communicate directly over a distance of up to 6 feet. In order to ensure reliable signal delivery over that distance, we enhanced receiver functionality in the following ways:

- Summing FFT Samples
- High-Frequency Bands

Averaging FFT Samples

The major challenge in transmitting over distance is the fact that the signal strength drops off proportionally to the square of distance while noise stays constant. However, white noise does not maintain power in a single frequency over an extended period of time. So, when we added up the noise over several FFT samples, we were able to recover the signal over distance.

High-Frequency Bands

Although we solved the problem of white noise by summing FFT samples, we experienced problems with constant room noise, such as ventilation and computer fans. This noise is constant in certain frequencies and gets amplified when the signals are added up. However, this noise is present primarily in lower frequencies. By moving our data bands into high frequencies we were able to avoid sources of noise.

Ad-Hoc Routing

To accomplish routing we designed and implemented GORAN (Goran[1] On-Demand Routing for Ad-Hoc Networks). We arrived at this protocol by considering the following high-level goals:

- Minimize transmission of routing control information
- Minimize network packet size

These goals reflect our attempt to reserve precious channel capacity for data transmission. In GORAN routing we minimize transmission of routing control information by forming a route on-demand when a transmitting computer requests one. To minimize packet size, we do not include the whole route in the packet transmitting the data, but rely on intermediate hops to forward the packet based on their route caches. The down side of GORAN is that it requires extra time when establishing a connection for the first time. We also made an assumption that the hosts are not mobile, however, GORAN would also perform well in a dynamic environment.

Route Discovery

Route discovery process requires three mechanisms:

- Route Request. If the source does not have the route to designation in its route cache, it broadcasts a route request (RREQ) packet to its neighbors. If a node is not the destination, it adds itself to the route in the packet, updates its route cache with the route in the packet so far and forwards the request to the neighbor. RREQ travels through the network until it reaches the destination. We utilize a sequence number to avoid loops.

[1] We name our system after Goran Ivanisevic who eventually won the Wimbledon Title after 13 tries.

- Route Reply. If the node receives a route request addressed to it, it forms a route reply (RREP) packet and sends it back to the source. At this point the reply has a complete route from source to destination. The intermediate nodes update their route caches again with the remaining path information. Each node on the route now knows how to forward data from the source to destination.

- Route Error. If a MAC layer cannot forward data, it reports a link failure to the network layer. The failed node forms a route error (RERR) packet and sends it back to the source of the data. Every intermediate node removes failed link from route cache and forwards error packet to next hop back to the source, which also updates its cache upon receipt.

Routing
Once the route is known, routing is accomplished based on the information in the route caches. The dynamic is illustrated in the figure below in the simple case of three hosts.

- The source creates a network packet with the destination address in the “Destination” field and hands it off to the MAC layer, specifying next hop from route cache as a MAC layer destination address.
- The intermediate node receives the packet, looks up next hop based on its routing cache, and passes it back to MAC layer with the next hop as the Mac layer destination address.
- The destination node receives the packet and passes it to the application layer

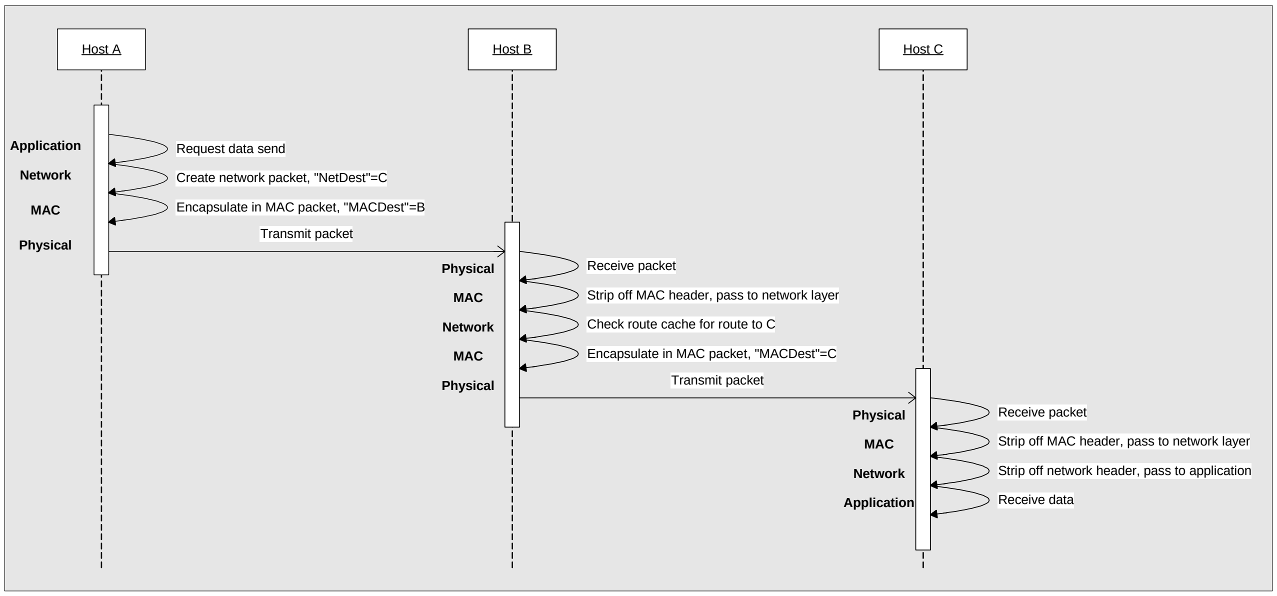

Host A
Host B
Host C
Application
Network
MAC
Physical
Request data send
Create network packet, "NetDest"=C
Encapsulate in MAC packet, "MACDest"=B
Transmit packet
Physical
MAC
Network
MAC
Physical
Receive packet
Strip off MAC header, pass to network layer
Check route cache for route to C
Encapsulate in MAC packet, "MACDest"=C
Transmit packet
Physical
MAC
Network
Application
Receive packet
Strip off MAC header, pass to network layer
Strip off network header, pass to application
Receive data

Figure 1: Data Routing Protocol

Although the illustration shows only three nodes, this protocol can scale to an arbitrary number of intermediate nodes.

VI. Phase 3 Design

Resource control was accomplished using the following principles:

- No overhead of MACA
- Parameterized maximum packet size
- Reduced overhead traffic
- Carrier sense at the receiver
- Fast transmission rates
- Random exponentially increasing back-off
- Fair back-off

We now explore these design principles in detail.

Sharing Air

In earlier phases we implemented carrier sense at the sender to detect channel availability. If channel was deemed idle, the information was sent immediately. Otherwise, a random back-off timer was set. This protocol did not address fairness, but it allowed for two nodes to "share air" in the basic sense.

In this phase we attempted to improve our channel control algorithm. First we considered MACA protocol, which was developed for packet radio networks. We quickly realized that MACA will not be extremely effective in our environment for several reasons:

Packet size must be small. The audible sound channel is inherently lossy, so when transmitting a long stream of data a bit eventually gets scrambled. We implemented error correction that takes care of one bit per byte losses, but several bits per byte losses are not uncommon. Thus, transmitting long packets is not practical; in our implementation we set our maximum packets size to 300 bits.

Packet loss rate is high. Even when using smaller packets, loss rate is significant (~10%).

Overhead of RTS/CTS is not justified for throughput. Based on packet size limitation, we chose to not implement MACA resource control mechanism because it is only useful when data packet size is much larger than the RTS/CTS overhead. In our case RTS/CTS packets would be on the order of 32 bits (addresses ~8, length ~8, control/sequence ~8, CRC ~8). In ideal case, overhead of RTS/CTS pair is 64 bits. Furthermore, with every RTS packet loss the overhead increases.

In idealized scenario when there were no collisions, CTS/RTS would not be necessary. As the probability of collisions increases, the overhead of RTS/CTS becomes increasingly more justifiable. We'd like to calculate the maximum rate of collisions per sec (X) for which the overhead of MACA is lower than the packet loss rate. We will use the following formula for our "back of the envelope" calculations.

Time to transmit w/o MACA < Time to transmit with MACA

Average number of bits required to transmit a packet w/o MACA N_{NM} is

N = Number of bits per packet = 300
R_{loss} = Rate of Loss = 0.1
R_{coll} = Rate of Collisions = X

$N_{NM} = N + N*R_{loss} + N*R_{coll}$

This means that on average we send the bits in the packet and some additional bits depending on the rate of loss and collisions.

Average number of bits required to transmit a packet N_{YM} is

N = Number of bits per packet = 300
N_{RTS} = Number of bits in overhead of RTS = 32
N_{CTS} = Number of bits in overhead of CTS = 32
R_{loss} = Rate of Loss = 0.1
R_{coll} = Rate of Collisions = X

$N_{YM} = (N+N_{RTS}+N_{CTS}) + (N+N_{RTS}+N_{CTS})*R_{loss} + (N_{RTS})*R_{coll}$

This means that on average we send the bits in the packet plus overhead RTS and CTS. We experience losses on all three components, but we experience collisions only on RTS.

It is easy to see that if R_{coll} is zero, N_{NM} is always more than N_{YM}, because of additional RTS/CTS overhead. However, as R_{coll} increases $N*R_{coll}$ becomes more significant and can justify the need for MACA. To calculate this threshold, we calculate X - Rate of Collisions (R_{coll}):

Time to transmit w/o MACA < Time to transmit with MACA

$[N + N*R_{loss} + N*R_{coll}]*R_{trans} < [(N + N_{RTS}+N_{CTS}) + (N+N_{RTS}+N_{CTS})*R_{loss} + (N_{RTS})*R_{coll}]*R_{trans}$

$[300 + 300*0.1 + 300*X]*R_{trans} < [(300 + 32 + 32) + 300 + 32 + 32)*0.1 + (32)*X]*R_{trans}$

$X < 23\%$

From above calculations, for rates of collisions under 23% our network achieves higher throughput than MACA. So for rates of collisions under 23%, RTS/CTS overhead offers no benefit in terms of throughput. For most realistic scenarios in a sparse network like ours is likely to be less than 23%, because packet sizes will vary and not necessarily be as large as the maximum packet size of 300 bits.

To ensure that our rate of collisions is low, we implemented several optimizations to minimize collisions, increase throughput and ensure fairness:

- Fast transmission rate (128 bits/sec) achieved by efficient signal processing and use of multiple data transfer channels
- Parameterized maximum packet size that can be adjusted to reduce collision rate
- No regular routing updates to reduce overhead traffic
- Carrier sense at the receiver to detect channel utilization
- Random exponentially increasing back-off to ensure turn taking
- Fair back-off implemented by pausing the retransmit timer instead of restarting it, so nodes that have been waiting longer have shorter retransmit counter times

Interoperability

In this phase we implemented the gateway responsible for tunneling packets from our network to other networks with completely different requirements. The gateway has the following features:

- Connects two heterogeneous networks
  - Our network (N1): On-Demand Source-like Routing
  - Connected network (N2): Distance-Vector Routing
- Application-level data tunneling, since there is no common network or transport interface
- Transparent to nodes on both networks
  - Requests N1 → N2: Gateway responds to route requests to any address in N2
  - Requests N2 → N1: Gateway periodically broadcasts itself as a node that has a path to any address in N1

Usability

To improve usability of our test application we implemented a GUI that has the following features in one parent window:

- Application/Network/MAC/Physical level message history
- Destination/Message/Send Window for message entry
- Automatically created network graph

VII. Implementation

We implemented our design using Java 2 Platform Standard Edition 5.0 programming language. The rationale for the choice was the availability of Java Sound libraries, which greatly simplified signal processing tasks of this project and allowed us to focus on implementing the networking features.

Software Structure

The following diagram shows the modules we implemented within each networking layer.

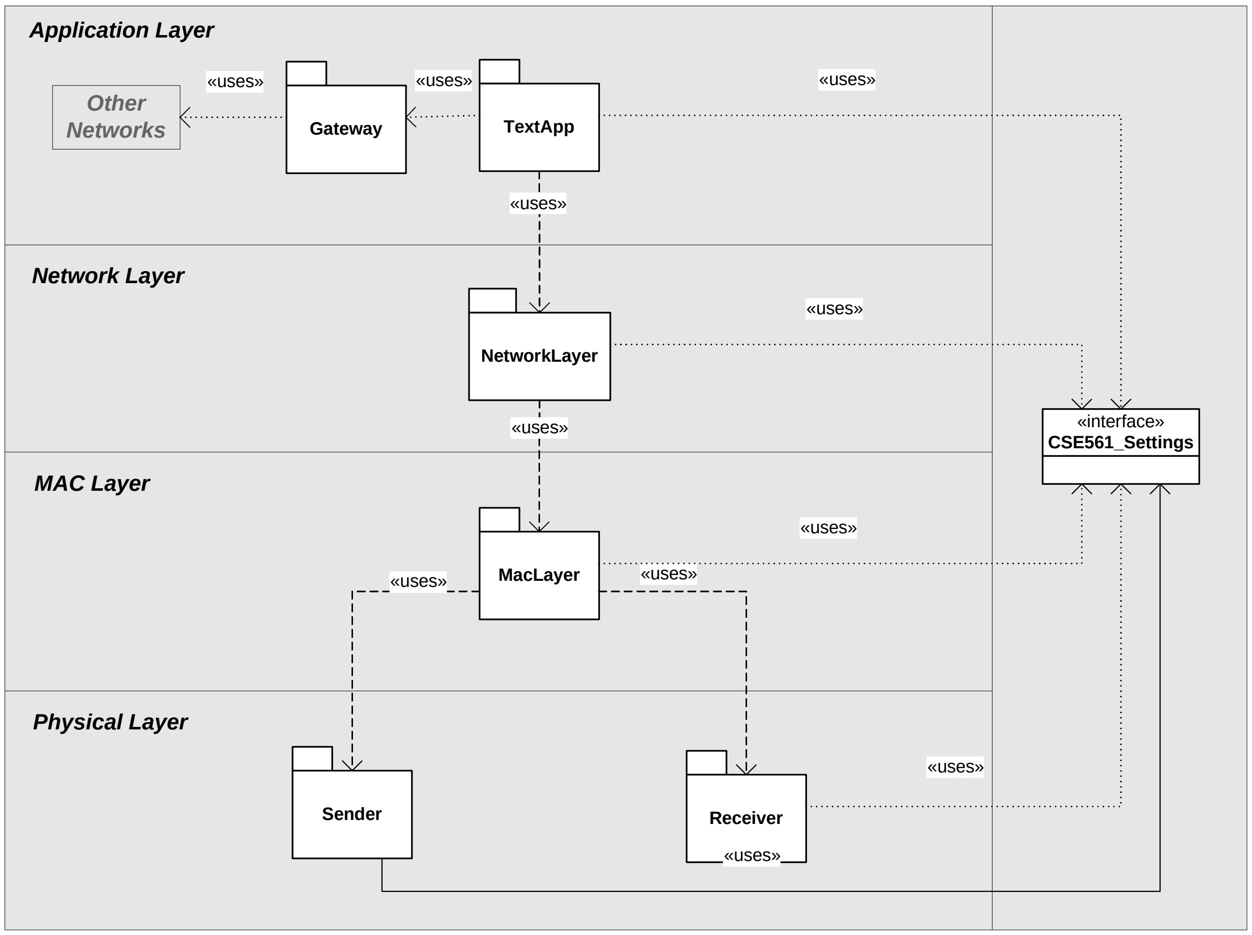


Figure 2: High-Level Implementation

Application Layer

Application layer is responsible for presenting the user with a GUI application that allows controlling the sending and receiving of data. We did not implement a transport layer, because we did not foresee implementing TCP-like functionality. However, this layer can easily be introduced in our modularized structure. Currently a specialized variant of an application layer takes care of gateway functionality described earlier.

Network Layer

On the sender side the network layer accepts bits of data from the application layer and passes them to the MAC layer. On the receiver side the network layer accepts bits from the MAC layer and passes them to the application. In addition ad hoc routing functionality is implemented at the Network Layer.

MAC Layer

On the sender side the MAC layer accepts bits from the network layer and packetizes them in preparation for sending. On the receiver side the MAC layer accepts bits from the physical layer, reconstructs the packet and passes it to the network layer. In addition, carrier sense, back-off and stop-and-wait retransmission mechanisms are implemented at this layer.

Physical Layer

On the sender side the physical layer accepts bit stream to send and converts it to a sound wave. On the receiver side the physical layer accepts amplitude signals from the sound card and reconstructs the bit stream by examining the frequency spectrum.

VIII. Testing

We have successfully tested Phases 1-3 requirements using the following procedure.

Procedure

To test the operation of basic communication and reliability, follow these steps:

1. Run TextApp.java <sender_address> on sender
2. Run TextApp.java <receiver_address> on receiver
3. Use GUI to send and receive messages

Basic Communication

Send one data packet to receiver. Expect link level acknowledgement. Retransmit if acknowledgement not received.

Routing

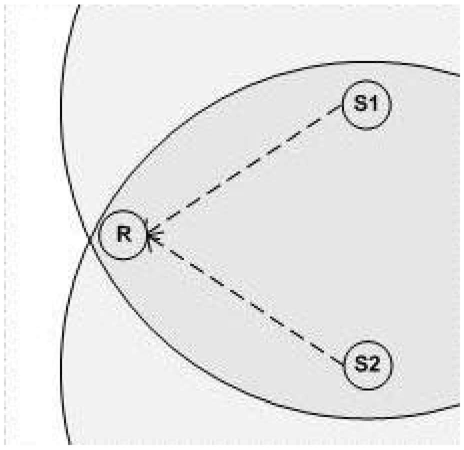
Send one data packet to receiver that is not within sender's range. Watch route discovery process and data transmission process.

Interoperability

Send one data packet to receiver that is on Mike's network. Watch the gateway pose as hosts on Mike's network by responding to route requests to his network addresses. Watch gateway forward packets to Mike's network hosts.

Congestion Control

Scenario 1 – Flooding One Receiver



Expected Results:

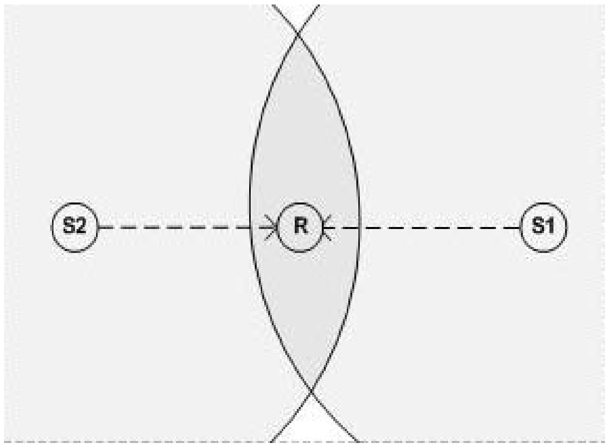
S1 and S2 will take turns sending packets to R.

Actual Results:

S1 and S2 indeed took turns sending packets to R.

Because in this case, S1 and S2 can detect when the other is transmitting, carries sense is sufficient to prevent collisions. Initial random back-off ensures with high probability that one of the senders will start first. Once that happens, the other sender waits until the first sender is finished. At this point, the waiting sender will have a shorter timeout (guaranteed by fair timeouts discussed before) and will transmit next, ensuring fairness.

Scenario 2 – Hidden Terminal



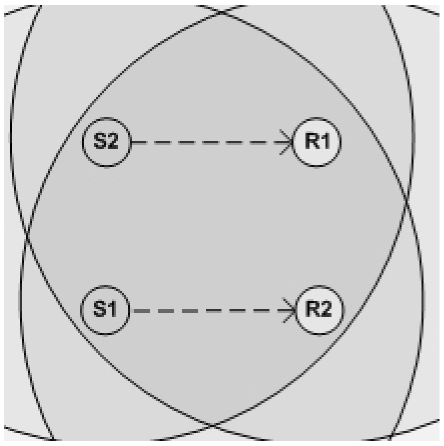
Expected Results:

S1 and S2 will take turns sending packets to R.

Actual Results:

S1 and S2 indeed took turns sending packets to R. Although S1 and S2 could not use carrier sense to prevent collisions, they could take turns sending packets by using exponentially increasing randomized timeouts. Fairness was assured because timeouts allowed the node that has been waiting the longest to send first. Efficiency was assured because although some collisions occurred initially, timeouts corresponded to length of time necessary to send packets and transmission succeeded on the second or third attempt.

Scenario 3 – Mesh



Expected Results:

S1 and S2 will take turns transmitting packets to R1 and R2.

Actual Results:

S1 and S2 indeed took turns sending packets to R1 and R2.

Because in this case, S1 and S2 can detect when the other is transmitting, carries sense is sufficient to prevent collisions. Initial random back-off ensures with high probability that one of the senders will start first. Once that happens, the other sender waits until the first sender is finished. At this point, the waiting sender will have a shorter timeout (guaranteed by fair timeouts discussed before) and will transmit next, ensuring fairness.

IX. Conclusion

We completed high level goals established for Phases 1-3 of the project.

Goal	Phase	Status	Notes
Basic Communication	1	Completed	Point to point communication
Reliability	1	Completed	Retransmissions, Stop & Wait
Robust Communication	2	Completed	Summing FFTs, High Frequencies
Ad Hoc Routing	2	Completed	Variant of Source Routing
Resource Control	3	Completed	Carrier Sense and Random Exponential Back-off
Efficiency/Fairness	3	Completed	High Data Rates / Fair Back-off

Table 1: Project Status