Voice over Sensor Networks

Rahul Mangharam[1] Anthony Rowe[1] Raj Rajkumar[1] Ryohei Suzuki[2]
[1]Dept. of Electrical & Computer Engineering
Carnegie Mellon University, U.S.A.
{rahulm, agr, raj}@ece.cmu.edu
[2]Ubiquitous Networking Laboratory
Tokyo Denki University, Japan
ryohei@unl.im.dendai.ac.jp

Abstract

Wireless sensor networks have traditionally focused on low duty-cycle applications where sensor data are reported periodically in the order of seconds or even longer. This is due to typically slow changes in physical variables, the need to keep node costs low and the goal of extending battery lifetime. However, there is a growing need to support real-time streaming of audio and/or low-rate video even in wireless sensor networks for use in emergency situations and short-term intruder detection. In this paper, we describe a real-time voice stream-capability in wireless sensor networks and summarize our deployment experiences of voice streaming across a large sensor network of FireFly nodes in an operational coal mine. FireFly is composed of several integrated layers including specialized low-cost hardware, a sensor network operating system, a real-time link layer and network scheduling. We are able to provide efficient support for applications with timing constraints by tightly coupling the network and task scheduling with hardware-based global time synchronization. We use this platform to support 2-way audio streaming concurrently with sensing tasks. For interactive voice, we investigate TDMA-based slot scheduling with balanced bi-directional latency while meeting audio timeliness requirements. Finally, we describe our experimental deployment of 42 nodes in a coal mine, and present measurements of the end-to-end throughput, jitter, packet loss and voice quality.

1. Introduction

Wireless sensor networks are composed of low-cost battery-operated nodes which communicate across one or more hops to at least one gateway. In order to keep costs low and maintain energy-efficient operation, nodes generally employ an 8-bit or 16-bit microcontroller and a low-rate short-range radio transceiver [1]. The limited processing power and network bandwidth available in such networks has traditionally restricted operation to applications with low duty-cycle such as infrequent sensing and monitoring, in-network data reduction and asynchronous operation [2][3][4]. While a majority of traditional sensor network applications focuses on passive sensing and reporting, there is a growing need to support real-time streaming for voice and low-rate video delivery in both mission-critical operations and in wide-area surveillance, particularly under emergency conditions and for intruder detection alerts. Such applications require relatively high bandwidth utilization, impose severe constraints on the end-to-end delay and require tight coordination between sensor nodes. The goal of this paper is to describe a real-time voice streaming capability in wireless sensor networks, and summarize our deployment experiences with voice streaming across a large sensor network in a coal mine. We describe the system-level decisions we made in hardware design for tight time synchronization, a TDMA-based media access protocol, the use of a real-time operating system for sensor nodes and packet scheduling for streaming voice across multiple hops.

We now describe an application domain which captures the requirements that we aim at satisfying. Over the past decade, there has been a surge of accidents in coal mines across the world. In most cases, miners are trapped several thousand feet below the surface for several hours while rescuers try to find their location and attempt to communicate with them. In January 2006, 13 miners were trapped for nearly two days in the Sago Coal Mine in West Virginia, USA. The miners were less than a few hundred feet from an escape route but were not aware of it. Similarly, in February 2006, in the Pasta de Conchos coal mine in Mexico, 65 miners were trapped more than 1 mile below the ground level. After more than 100 hours of rescue attempts, the authorities were still unable to locate or establish communication with the miners. In both cases, the prevalent wired communication systems were destroyed when a portion of the mine collapsed and there was no way to re-establish connection to the affected areas. Fatalities often result in such mining incidents.

The normal practice to check the status of the trapped miners is to drill a narrow hole (of 1-2 inch radius) from the surface to a mine tunnel and drop a microphone, camera and air quality sensors at different locations around the disaster area. This method provides limited access to the affected region as medium-sized mines may span several miles across. Another method of communicating to the miners is by installing a loop antenna that is several miles long, over the surface of the mine. This scheme uses a low-frequency transmitter on the surface to send one-way broadcasts of short text messages and is unable to get feedback about the status or location from the miners below.

Our group was invited to investigate the use of wireless sensor nodes to track miners and to evaluate their viability as an end-to-end rescue communication network for miners during an incident. We proposed the establishment of a self-healing wireless network in such mine-like environments to

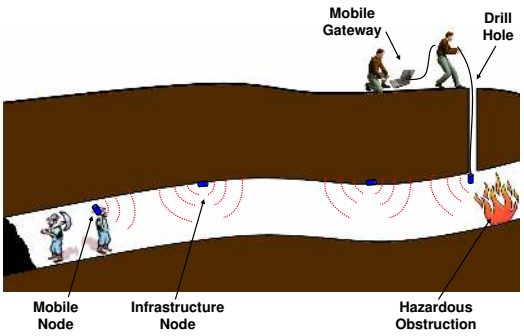


Figure 1. Rescue Sensor Network in Coal Mine

maintain communication in the remaining connected network. As shown in Figure 1, if a wireless node was lowered through the drill-hole, it could re-establish communications with the remaining network and initiate two-way communication with the miners. In addition, the miners would be able to leave broadcast voice mail-type messages and allow it to propagate to all nodes in the remaining network. It is important to note that during normal operation, the network's primary task is to track miners and record environmental data.

In order to keep normal network maintenance costs low, it is necessary to meet the following design goals:

1. All nodes are to be battery-powered.
2. Nodes must have a predictable lifetime of at least one to two years under normal operation.
3. Nodes must provision continuous voice communication for at least one week with fully-charged batteries.
4. Voice communications must include two-way interactive calling, one-way "push-to-talk" voice messaging and support for store and broadcast voicemail messaging.
5. The network must be able to tolerate topology changes and self-heal to maintain connectivity after a network partition.

The two fundamental challenges in delivering delay-bounded service in sensor networks are (a) coordinating transmission so that all active nodes communicate in a tightly synchronized manner and (b) ensuring all transmissions are collision-free. Time synchronization is important because it can be used to pack the activity of all the nodes so that they may maximize a common sleep interval between activities. Furthermore, it can be used to provide guarantees on timeliness, throughput and network lifetime for end-to-end communication. In this paper, we focus on the first four goals of voice streaming during steady-state network operation.

1.1. Overview of Sensor Network Streaming

We use the FireFly sensor network platform to implement on-board audio sampling, ADPCM encoding, packet transmission and forwarding. Each FireFly node has an 8-bit microcontroller and two radios: an IEEE 802.15.4 transceiver for multi-hop data communication and a low-power AM receiver for global time synchronization. Each node receives a short (e.g. 50μs) periodic pulse from an AM transmitter once every few seconds (e.g. 6 sec). The AM time sync pulse is broadcast to all nodes in the network and provides a global time reference. This design enables each node to operate in a tightly coupled local and networked synchronized regime with nominal overhead.

Multiple applications such as audio processing, localization and sensor updates are managed by the Nano-RK real-time sensor operating system [5]. As shown in Figure 2, the Nano-RK kernel includes the RT-Link real-time link protocol [6] which uses hardware-based time synchronization to mark the beginning of a new TDMA communication cycle. RT-Link supports fixed and mobile nodes. RT-Link also supports both out-of-band hardware-based global time sync and in-band software-based time sync. Audio packet transmission and forwarding are executed in explicitly scheduled time slots which are assigned by the gateway and maintained by the network task in Nano-RK.

1.2. Organization of this Paper

We provide a review of our hardware platform and our real-time link protocol in Section 3. We describe our implementation of various voice codecs for an 8-bit microcontroller in Section 4. In order to facilitate multiple delay-sensitive tasks on each node, we describe the use of the Nano-RK RTOS and network scheduling in Section 5. A performance study is presented in Section 6. Concluding remarks are provided in Section 7.

2. Related Work

Link and network support for real-time communications over sensor networks have attracted much attention in the recent years. Constraints on computing power, bandwidth, memory and energy supply in sensor networks make the problem of delivering timeliness guarantees across multiple hops especially challenging. In [7], Abdelzaher presents the capacity bounds on how much real-time data a sensor network can transfer by the imposed deadlines. He derives a sufficient schedulability condition for a class of fixed-priority packet scheduling algorithms and provides a theoretical basis for capacity planning. Several media access schemes have been

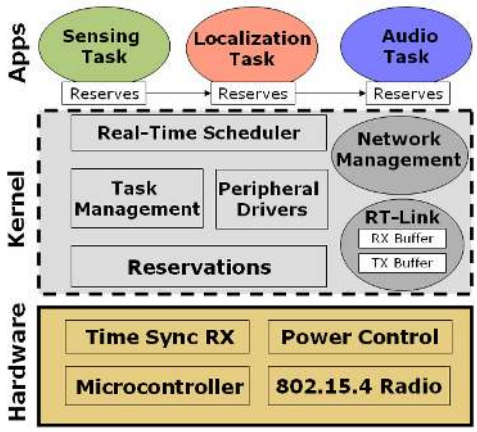


Figure 2. Time synchronization tightly couples the real-time operating system, link and network protocols in the FireFly sensor network platform

proposed for real-time communication over sensor networks, with the goal of bounding the end-to-end delay. In [8], the authors present velocity-monotonic scheduling that accounts for both time and distance in large-scale networks. Simulation studies show that such a scheme is effective in minimizing the deadline-miss ratios in multi-hop sensor networks. Finally, both [9] and [10] present simulation studies on resource reservation and routing protocols for applications with end-to-end timeliness constraints.

TDMA protocols such as TRAMA [11] and LMAC [12] are able to communicate between node pairs in dedicated time slots. Both protocols assume the provision of global time synchronization but consider support for it to be an orthogonal problem. FireFly integrates time synchronization within the link protocol and also in the hardware specification. FireFly has been inspired by dual-radio systems such as [13, 14] used for low-power wake-up. However, neither system has been used for time-synchronized operation. Several in-band software-based time-synchronization schemes such as RBS [15], TPSN [16] and FTSP [17] have been proposed and provide good accuracy. In [18], Zhao provides experimental evidence showing that nodes participating in multi-hop communications in an indoor environment routinely suffer link error rate over 50% even when the receive signal strength is above the sensitivity threshold. This limits the diffusion of in-band time synchronization updates and hence reduces the network performance. RT-Link employs an out-of-band time-synchronization mechanism which also globally synchronizes all nodes and is less vulnerable than the above schemes.

Real-time voice encoding on 8-bit microcontrollers is a severe challenge due to the limited amounts of processing power, random access memory and bandwidth available in low-cost sensor nodes. A description of ADPCM decoding using a PIC microcontroller is provided in [19]. Similarly, [20] describes an off-line method to decode ADPCM on an Atmel microcontroller but requires the use of flash memory. Our implementation has drawn from these experiences but encodes the raw audio samples on-line and does not require writing samples to flash.

To the best of our knowledge, the FireFly platform is one of the first low-cost and low-power systems capable of real-time streaming across multiple hops in a wireless sensor network. The combination of global hardware-based time synchronization, a TDMA link layer capable of collision-free communication, multiple task scheduling on each node, implementation of low-rate low-complexity audio compression and network scheduling provide a stable framework for real-time streaming.

3. FireFly Sensor Platform

In this section, we present a review of the FireFly hardware, Nano-RK RTOS and the RT-Link protocol on each node.

3.1. FireFly Sensor Hardware

At Carnegie Mellon, we have developed a low-cost low-power hardware platform called FireFly as shown in Figure 3. Firefly uses an Atmel ATmega32 [21] 8-bit Harvard architecture microcontroller with 2KB of RAM and

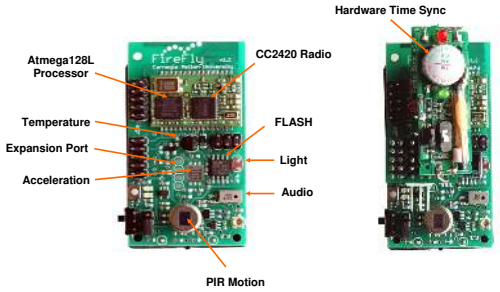


Figure 3. FireFly nodes with multiple sensors, an 802.15.4 transceiver and an add-on AM receiver

32KB of ROM along with Chipcon's CC2420 IEEE 802.15.4 standard-compliant radio transceiver [22] for communication. The board has light, temperature, audio, passive infrared motion, dual axis acceleration and voltage monitoring built in. Built-in hardware support is available for global time-synchronization and for control over peripheral power. We use the individual sensor power control to aid in Nano-RK's resource reservation enforcement. The maximum packet size supported by 802.15.4 is 128 bytes and the maximum raw data rate is 250Kbps.

To support voice sampling, we use a low-cost MEMS-based microphone [23]. The microphone has a sensitivity of -26dB and a flat frequency response up to 8KHz. The microphone has a built-in amplifier and consumes 0.1mA at 3V. The ATmega32 microcontroller has a 10-bit analog-to-digital converter and delivers good raw voice quality.

3.2. Nano-RK Real-Time OS

Nano-RK, described in [5], is a reservation-based real-time operating system (RTOS) with multi-hop networking support for use in wireless sensor networks, and runs on the FireFly sensor nodes. It supports fixed-priority preemptive multitasking for ensuring that task deadlines are met, along with support for and enforcement of CPU and network bandwidth reservations. Tasks can specify their resource demands and the operating system provides timely, guaranteed and controlled access to CPU cycles and network packets in resource-constrained embedded sensor environments. It also supports the concept of virtual energy reservations that allows the OS to enforce energy budgets associated with a sensing task by controlling resource accesses. A lightweight wireless networking stack (to be discussed in the next subsection) supports packet forwarding, routing and TDMA-based network scheduling. Our experience shows that a light-weight embedded resource kernel (RK) with rich functionality and timing support is practical on sensor nodes.

3.3. RT-Link Protocol Design

RT-Link, described in [6], is a TDMA-based link layer protocol for multi-hop sensor networks and provides predictability in throughput, latency and energy consumption. All packet exchanges occur in well-defined time slots. Global time synchronization is provided to all fixed nodes by a robust and low-cost out-of-band channel. In-band time synchronization is also supported for mobile nodes and for fixed nodes that are not within the range of the global time sync broadcast.

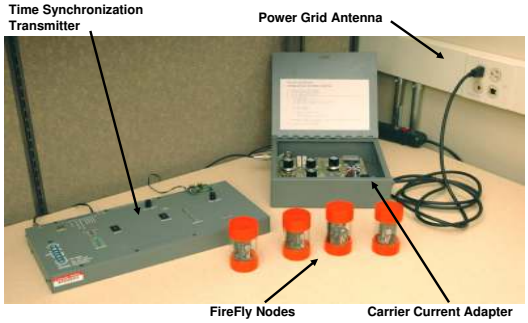


Figure 4. Carrier current-based AM radio transmitter for global time synchronization

3.3.1. Hardware-based Global Time Sync

RT-Link supports two node types: *fixed* and *mobile*. The fixed nodes have an add-on time sync module which is normally a low-power radio receiver designed to detect a periodic out-of-band global signal. In our implementation, we designed an AM/FM time sync module for indoor operation and an atomic clock receiver for outdoors. For indoors, as shown in Figure 4, we use a carrier-current AM transmitter [24] which plugs into an ordinary power outlet and uses the building's power grid as an AM antenna to radiate the time sync pulse. We feed an atomic clock pulse as the input to the AM transmitter to provide the same synchronization regime for both indoors and outdoors. The time sync module detects the periodic sync pulse and triggers an input pin in the microcontroller which updates the local time. Our AM receiver currently draws 5mA while the 802.15.4 transceiver consumes 20mA during packet reception. The relatively low energy overhead and robustness of the AM signal make it our preferred mode of time synchronization in multi-hop sensor networks. In our experiments, a single AM transmitter has been used to provide global time synchronization with a sub-20μs accuracy to a large 8-story campus building.

As shown in Figure 5, the time sync pulse marks the beginning of a finely slotted data communication period. The communication period is defined as a fixed-length cycle and is composed of multiple frames. Each frame is divided into multiple slots, where a slot duration is the time required to transmit a maximum sized packet. RT-Link supports two kinds of slots: *Scheduled Slots* (SS) within which nodes are assigned specific transmit and receive time slots and (b) a series of unscheduled or *Contention Slots* (CS) where nodes, which are

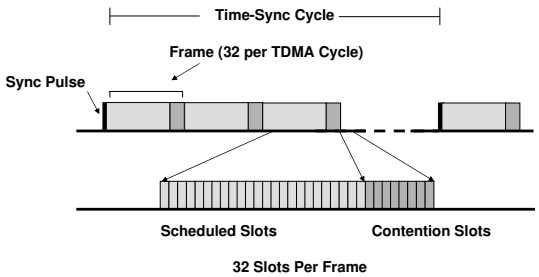


Figure 5. RT-Link TDMA-based protocol with scheduled and contention slots

not assigned slots in the SS, select a transmit slot at random as in slotted Aloha. Nodes operating in SS are provided timeliness guarantees as they are granted exclusive access of the shared channel and enjoy the privilege of interference-free and hence collision-free communication. While the support of SS and CS are similar to what is available in IEEE 802.15.4, RT-Link is designed for operation across multi-hops. After an active slot is complete, the node schedules its timer to wake up just before the expected time of next active slot and promptly switches to sleep mode.

In our default implementation for voice delivery, each cycle consists of 32 frames and each frame consists of 32 6ms slots. Thus, the cycle duration is 6.144sec and nodes can choose one or more slots per frame up to a maximum of 1024 slots every cycle. Each 6ms slot includes 4ms to transmit 128 bytes at a rate of 250Kbps, a 2ms duration for packet aggregation for forwarding, guard time for synchronization jitter and for the time it takes to write to the cc2420 FIFO. Given the 2ms overhead, we note that this is the highest rate of pipelined and in-network aggregation achievable with the cc2420 transceiver. The common packet header includes a 32-bit transmit and 32-bit receive bit-mask to indicate during which slots of a node is active. RT-Link supports 5 packet types including *HELLO, SCHEDULE, DATA, ROUTE* and *ERROR*. The RT-Link header is 16 bytes large and yields a 112 byte effective payload per packet.

3.3.2. Software-based In-band Time Sync

For fixed nodes not in the range of the AM broadcast and for mobile nodes which do not have the AM receiver, it is necessary to support in-band software-based sync over 802.15.4. In the coal mine testbed, our system had to support an in-band sync mechanism because the power lines over which the AM carrier current can be distributed are prone to getting disconnected when an explosion or disaster situation occurs. FireFly supports in-band time sync by adding the absolute slot number (out of 1024 slots) in the packet header. Any node listening to its neighbors will be able to synchronize upon a reception and then transmit. By default, nodes do not transmit unless they have received a sync message. If a node has not received an out-of-band sync pulse for more than 5 cycles, it switches its transceiver on and activates in-band time sync. Mobile nodes do not have a fixed neighborhood and hence cannot be assigned a static schedule. In-band time sync synchronizes mobile nodes and helps them find the next group of contention slots where they can transmit. While in-band time sync is more expensive in terms of energy consumption than hardware sync because nodes have to wait until they hear the first neighbor message, it provides a practical and scalable mechanism to support additional node types.

3.3.3. Multi-Rate Support

With RT-Link, nodes can be scheduled on explicit slots and on slots based on the desired rate of communication. In order to flexibly use a global TDMA schedule, RT-Link supports six different rates based on a logarithmic scale as shown in Table 1. Each node can be scheduled based on how often it is assigned a slot per frame. A node's transmit and receive slot rate is described by a 3-bit rate field in the packet header

Frame Rate Index	Slot Interval	Frames/ Cycle	Max. Goodput (Kbps)
0	-	0	0
1	1	32	149.3
2	2	16	74.6
3	4	8	37.3
4	8	4	18.6
5	16	2	9.3
6	32	1	4.6

Table 1. RT-Link Multi-rate support.

and is assigned its slot schedule via the SCHEDULE frame. The highest frame rate supported is *rate 1* where the node transmits on every frame and provides the maximum goodput of 149.3Kbps if all slots are set in each active frame. Nodes operating on *rate 2* transmit or receive every other frame or 16 frames per cycle. If all slots are set in each active frame, this results in 512 active slots per 6.114ms cycle and have a goodput of 74.6Kbps.

4. Voice over RT-Link Protocol

In order to deliver voice as a primary means of communication under emergency conditions, we needed to support 2-way interactive voice, 1-way push-to-talk messaging and a voice mail application where a user could broadcast a voice snippet at moderate speed to nodes on the periphery of the network. We focus on 2-way voice communication as it is the most stringent in terms of end-to-end delay and throughput requirements. We choose the Adaptive Differential Pulse Code Modulation (ADPCM) waveform codec as it provided us with a set of low transmission data rates, was easy to implement on an 8-bit fixed-point architecture and has low processing overhead. ADPCM is simpler than advanced low bit-rate vocoders and can complete encoding and decoding in a relatively short time. The principle of ADPCM is to predict the current signal value from the previous values and to transmit only the difference between the real and the predicted value. As the dynamic range of this difference is small, we are able to get a compression ratio of 8:1. The microphone was sampled at 4KHz and output an 8-bit value which was compressed to a 4-bit, 3-bit or 2-bit ADPCM code. This enabled us to reduce a full-rate uncompressed voice stream of 64Kbps into a 16, 12 or 8Kbps compressed stream. We chose to sample the microphone at a rate lower than the normal 8KHz because it reduced the data rate by 50% and did not degrade the voice quality significantly.

4.1. FireFly Voice Codecs

FireFly currently supports raw audio sampling, 16Kbps, 12Kbps and 8Kbps ADPCM encoding. Each ADPCM unit is encoded at the time the microphone is sampled. Nano-RK maintains a double buffer to store the encoded data between transmission intervals. Table 2 lists the number of concurrent unidirectional raw audio and ADPCM streams supported between a node and the gateway across multiple hops. We compare its performance with GSM as it is an efficient codec that can also be implemented in 8-bit fixed point with moderate processing overhead. The number of unique slots which repeat are given by 2^r, where r is the RT-Link rate. For example, *rate 3* features an 8 slot repetition interval. A node may be only scheduled to transmit on slots that are separated by at least 3 slots so as to facilitate pipelining in the presence of the hidden terminals [25].

For *rates 1* and *2*, where a node transmits on every or every other slot respectively, only single-hop communication is possible. For *rate 1*, every 6ms slot is used to forward a voice packet to a receiver. In 6ms, 24 bytes of raw audio or 12 bytes of ADPCM-1 or 6 bytes of ADPCM-3 are captured. ADPCM-1 is able to pack 9 concurrent flows in the 112-byte payload every 6ms. As voice can be pipelined along a chain of nodes when at least 3 unique slots are available [25], ADPCM-1, ADPCM-2, ADPCM-2 and GSM-1 are able to support bi-directional voice across multiple hops. For *rate 3*, a node along a chain transmits only once every 4 slots and hence captures 24ms of voice data. At *rate 4*, a node along a chain may transmit once every 4 slots for bi-directional streams or once every 8 slots for unidirectional streams. With the network schedules used in Section 7, a node transmits a packet every 4 slots for bi-directional traffic. As each node is assigned a slot unique in its 2-hop neighborhood, both its neighbors are in receive mode during its transmit slot. Thus a node is able to concatenate both neighbors data in one packet and send it as a single transmission.

4.2. Voice Quality Trade-offs

As the RT-Link slot interval increases, fewer concurrent flows are supported due to the large size of captured audio over a longer time interval. *Rates 3* and *4* are very important because they support pipelining of data along multiple hops. *Rate 4* features an 8-slot cycle and thus 2 good quality ADPCM-1 flows can facilitate 2-way conversation or 2 lower quality ADPCM-3 streams with redundant packets can be supported. In Figure 6, we see this trade-off between higher quality streams and lower-quality streams but with re-

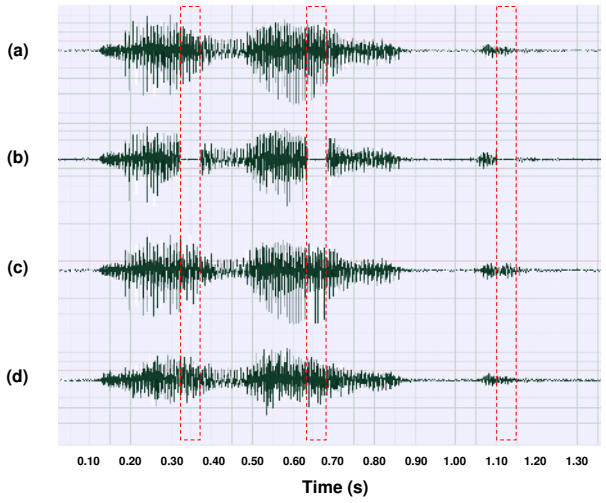


Figure 6. Error Concealment schemes:(a) Raw 4Khz 8-bit audio stream, (b) 4:1 ADPCM stream with 3 dropped packets (10% packet loss) and replaced with background noise, (c) 4:1 ADPCM with dropped packets replaced with the previous packet, (d) 8:1 ADPCM with redundancy.

RT-Link Slot Rate	Raw Audio 32Kbps	ADPCM-1 16Kbps	GSM-1 13Kbps	ADPCM-2 12Kbps	ADPCM-3 8Kbps	GSM-2 7Kbps	Avg. Hop Delay	Voice Reliability
1	4	9	11	12	18	21	6ms	Single
2	2	4	5	6	9	10	12ms	Single
3	1	2	2	3	4	5	24ms	Single
4	1	2	2	2	4	4	24ms	Double
5	0	0	0	0	4	4	48ms	Double

Table 2. Number of concurrent voice streams supported over RT-Link

dundant data. Figure 6(a) shows the original 4KHz sampled voice signal and Figure 6(b) shows the good quality ADPCM-1 with three lost packets. As detailed in [26], error concealment by replacing silence due to dropped packets with background noise is more effective than forward error correction, source coding or interleaving in delivering reasonable quality interactive voice even with lost packets. Figure 6(c) shows the same signal with dropped packets concealed by inserting the previous packet in place of the silence. This scheme results in a slight echo but the voice is audible. Finally, in Figure 6(d), we observe a lower amplitude output of low quality ADPCM-3 with packet repetition. While the voice quality is lower than Figure 6(c), there are no echoes or partial words.

To summarize the above discussion, there is a trade-off between higher quality encoding or low quality audio with redundant packets. As we will see in Figure 7, a majority of packet errors we observed were single packet errors with relatively few burst errors composed of multiple dropped packets. Thus, our recommendation is to use *rate 4* with ADPCM-1 for 2-way interactive voice when the observed error rate is low and switch to lower rate ADPCM-3 if the end-to-end error rate exceeds 10%.

4.3. Voice Call Signaling

In order for a mobile node to connect to the network and establish a call, we incorporated a simple voice call signaling handshake. Under normal conditions, all nodes in the sensor network operate at a low duty cycle and are active for the first slot in their sequence (as in Figure 10(a)) with *frame rate 5* (active only in the first and sixteenth frame). As RT-Link supports Low Power Listening [6], the cost for a node to wake up and listen 250μs for activity on an active slot is nominal. In each transmission, the fixed infrastructure nodes broadcast a *HELLO* message with their receive slots and hop distance from the gateway. A mobile node waits for one cycle before issuing a *CONNECT* message in the receive slot of the closest infrastructure node. This may be determined by choosing the node with the lowest hop distance and with a receive signal to noise ratio above the -85dBm stable threshold.

Once an infrastructure node receives a call *CONNECT* message, it forwards it to its upstream neighbor. There is an average delay of 1/4 cycle at every hop. This delay can be reduced by operating the nodes at a higher duty cycle under normal conditions and thus trades energy consumption for lower call setup latency. If a particular node en-route to the gateway is unable to honor the *CONNECT* request, it responds with a CALL_REJECT message and a reason code. If the message reaches the gateway, the gateway issues a *CALL_ACCEPT* to the mobile node and requests all nodes along the path to operate at the desired rate based on Table 2, from the beginning of the next cycle. Following this, the call is established and the mobile node will remain connected for the lease duration.

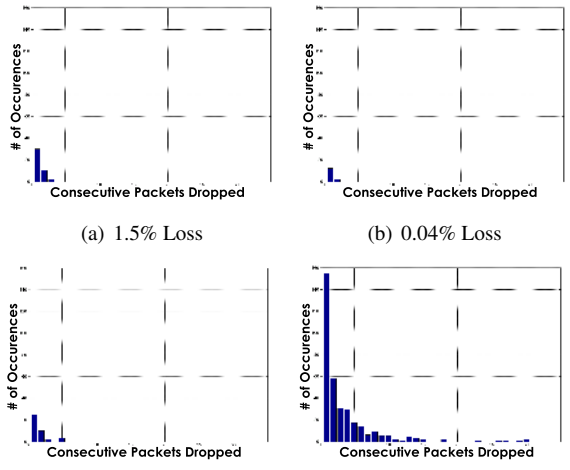
5. Node and Network Scheduling

In this section, we first discuss task scheduling on each node and then describe communications scheduling.

5.1. Task Scheduling with Nano-RK

While our primary goal is to facilitate coordinated transmission and reception in 6ms slots over the 802.15.4 interface, this has to be executed while sampling the microphone at 4KHz and tending to other tasks such as sensor updates and reporting, location tracking and real-time control of actuators. In order to ensure all tasks are executed in a timely manner, we use Nano-RK RTOS [5] for the FireFly sensor node. Nano-RK supports the classical operating system multitasking abstractions allowing multiple concurrent threads of execution. Rate-Monotonic Scheduling is used along with the Priority Ceiling Protocol to enforce task timeliness and deadlock-free synchronization. Nano-RK uses a novel energy-aware time management scheme that provides fine-grained time support while reducing idle energy costs. Nano-RK along with the RT-Link protocol requires 1KB of RAM and 10KB of ROM.

Our mining scenario consists of five tasks: a sensing task, audio task, localization task, network management task and link layer task. The link layer task described in [6] is responsible for managing individual link TDMA communication. The link layer sends signals to tasks and applications alerting them



(a) 1.5% Loss (b) 0.04% Loss

(c) 2.1% Loss (d) 52.3% Loss

Figure 7. Packet Loss Clustering At Four Points in a Multi-hop Chain of Nodes Streaming Audio

when packets are sent or received. The network management task is responsible for coordinating voice call setup, TDMA slot scheduling, mode changes and periodic network status updates. The audio task runs an audio driver (an interrupt routine) as well as the ADPCM compression algorithm. In our implementation, the sensing and localization tasks simply sampled RSSI values of mobile nodes and recorded sensor values. In a mature system, these tasks would be responsible for event detection processing and running in-network localization algorithms. In our configuration, the link layer is the highest priority task, followed by the audio driver, the network management task and then the sensing and localization tasks.

5.2. Network Scheduling

Given a sensor network topology, our goal is to schedule at least one interactive bi-directional audio stream from any connected point in the network to the gateway while unobtrusively allowing sensing tasks to continue normal network operations. Given a connected graph $G = \{V, E\}$, we find a schedule such that from any V to the root node, there is a path composed of nodes each of which transmits every n slots. n defines the data rate of the network and hence governs which audio encoding schemes are best suited for the system. To accommodate interactive voice, we must ensure that packet latency is symmetric between upstream and downstream communications and that packets arrive within acceptable timeliness bounds. The end-to-end latency of an audio stream is a function of the TDMA slot rate as well as the number of hops. Interactive voice requires an end-to-end latency of 250ms or less, beyond which users notice a drop in interaction quality. The goal for voice scheduling is therefore to minimize the number of unique slots to maximize n and to ensure the ordering of the slots results in balanced end-to-end delay in both directions of the flow.

Scheduling of the network is performed in two phases to cater to voice streaming and simultaneously to other network applications. First, time slots for audio streams are reserved. Next, the remaining slots are used to schedule lower data rate tasks such as sensor data reporting. It is important to schedule the audio streams first since these require strict latency bounds. Since our system initially only requires a single audio stream at a time, the two-hop coloring constraint associated with arbitrary communication scheduling is relaxed. In Figure 8(a), we see that for sensor data aggregation and forwarding to avoid the hidden terminal problem and be collision-free, each node requires a slot assignment which is unique in its 2-hop range. However, in Figure 8(b), we see that if only one voice connection is required at a time, the system requires a single upstream flow and nodes at each level in the tree can utilize the same schedule. This is a desirable property as it reduces the degree of the graph to that of a chain of nodes (i.e. 3). A network with a high-degree graph would quickly consume all available TDMA slots. As highlighted in Table 2, if a path is scheduled for a flow then multiple concurrent streams can be scheduled for slot rates *1* through *3* and redundant streams can be scheduled for slot rates *4* and *5*. In our deployment within the coal mine, as mining groups are few and far between, we scheduled the network to support a single end-to-end voice stream from any point in the network to the gateway.

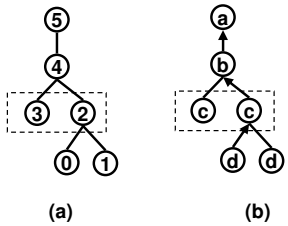


Figure 8. Slot assignment for (a) sensor sampling (with minimum latency to the gateway) and (b) simple streaming (with minimum latency for a single flow to the gateway)

The first step in creating the audio streaming schedule is to form a spanning tree across the network with the root located at the gateway. Based on the slot rate, a schedule is repeated down each linear chain in the network using a breadth first search. Since each path through the tree can be treated independently, all nodes at a common depth (>1) can be given identical schedules. Table 6 shows a sample collision-free scheduling pattern for an audio stream requiring a transmission every eight slots. The slot assignment may also be presented as a variant of a graph-coloring problem where a unique slot number is a unique color assigned to a node. Figure 9 shows how different scheduling schemes can adversely affect latency in each flow direction. Figure 9 (a) shows the minimum color schedule for a linear chain with a worst case delay of 31 slots per hop, (b) shows the minimal upstream latency coloring for sensor data collection tasks with a minimum upstream delay of 1 slot, and (c) shows our balanced schedule for bi-directional voice communication with symmetric delay of 4 slots in either direction. Next to each node, is an arrow indicating the direction of the flow and the number of slots of latency associated with the next hop. A change from a higher slot value to a lower slot value must wait for the next TDMA frame and hence may have a large delay. We observe that for high end-to-end throughput, minimizing the number of unique slots is essential. The minimum node color schedule in Figure 9(a) delivers the maximum end-to-end throughput for a chain of nodes, i.e. 1/3 the link data rate. Secondly, we see that for delay-sensitive applications, ordering of the slots is just as important as minimizing the colors. As seen in Figure 9 (c), we use an 8-slot sequence but achieve a lower end-to-end delay in both directions that uses fewer colors. Ordering the slots to balance the bi-directional end-to-end latency improves the voice performance significantly.

After the audio streaming slots have been reserved, the remaining slots are used to schedule sensing activities. As described in [6], we use a breadth first search with 2-hop graph coloring constraint and a slot reordering scheme that aims to minimize for uplink latency in order to periodically collect sensor data from all nodes.

6. Voice Quality & Network Performance Study

We now present and evaluation of the single-hop and multi-hop voice quality and robustness of the different AD-

MOS	Raw Audio 8-bit	ADPCM-1 4-bit	ADPCM-3 2-bit	ADPCM-1 Multi-hop	ADPCM-3 Multi-hop	ADPCM-3r Multi-hop
8KHz	-	3.2	3.0	2.8	2.6	2.9
4KHz	3.6	3.0	2.9	2.7	2.6	2.9

Table 3. Comparison of Mean Opinion Score

PCM encoding rates. In addition, the overall energy consumption of the system was measured to obtain an estimate of the stand-by and talk time of the network.

6.1. Voice Quality Evaluation

In controlled environments outside of the mine, we found that the system performed with below 3% packet loss per hop. Sending redundant data in separate packets allowed for more robust end-to-end voice transfers with improved quality. Figure 7 shows the distribution of packet loss clustering at four different hops along an eight hop series of nodes inside the coal mine. The end-to-end latency across the eight hops between when audio was sampled and when the playback occured was just under 200ms. Each hop maintained the expected rate with a four slot average latency of 24ms. We found that while the mine corridor is clear of obstructions the wireless channel shows few packet drops. In some situations, when a machine passes the narrow corridor, we observed temporary packet loss rates as high as 50%. Under these circumstances, packet drops are heavily clustered making error concealment or recovery difficult. Since occupancy inside a coal mine is relatively sparse (usually less than 5 groups) compared to the mine's size, clear paths are quite common. The mesh nature of sensor networks can ameliorate broken links by using alternative paths.

In order to measure the end-to-end performance of voice over our network, we asked a group of five testers to evaluate recorded audio samples. The group was asked to listen to samples of ADPCM-1 and ADPCM-3 with 4KHz and 8KHz sampling rates. In addition, we recorded ADPCM samples across 4 hops (with average packet error rate of 3%). Based on the group's rating after listening to the set of samples thrice, we computed the mean opinion score (MOS) of each audio sample (see Table 3). A MOS rating of 5 signifies excellent quality, a rating of 3 corresponds to fair quality with perceptible distortion that is slightly annoying and a rating of 2 corresponds to voice with annoying distortion but not objectionable. We are able to differentiate the speaker for MOS ratings of 3.0 and above. It was possible to differentiate between male and female for MOS ratings of 2.0 and above. Overall, although the voice with rating below 2.6 sounded synthesized, the content was easily perceptible.

For single hop communication, we observe that ADPCM-1 is only slightly better than ADPCM-3 even though it requires twice the data rate. For multi-hop, both ADPCM-1 and ADPCM-3 demonstrate lower quality but the performance of ADPCM-3r (with packet redundancy) is comparable to that of single-hop. We recommend the use of ADPCM-1 if the observed packet error rate is low and ADPCM-3r for multi-hop operation.

6.2. Energy Analysis

Table 4 shows the time and energy consumption of the various processing elements involved with the audio streaming. At a 4KHz sampling rate, 48 bytes of new data are generated using 2-bit ADPCM compression every 8 TDMA slots. The node consumes $0.9\mu J$ per byte to sample and compress an audio value and about $1.6\mu J$ per byte to transmit this compressed data (4 samples per byte). Without compression, each sample must be transmitted over the network consuming on average $1.65\mu J$ per sample. With 2-bit ADPCM compression, each audio sample including compression and transmission consumes on average $1.17\mu J$ per sample resulting in a 29% energy saving when compared to transmission of raw audio data.

Table 5 shows the predicted lifetime based on measured current drawn from the board. The boards typically draw on average .22mA at 3 volts while idle and 7.3mA while actively streaming every 8 slots. Using AA batteries, the voice streaming lifetime (i.e. talk time) reaches our requirement of 1 week. However, the overall system lifetime (i.e. standby time) of 1.45 years could be longer. Using two D-cell batteries, the idle system would theoretically last 8.8 years but is bounded by the battery shelf life of 5 years. We thus use D-cells for infrastructure nodes where wire line power is unavailable and AA batteries for hand-held units. At 5 years, this easily meets our requirement (doubled just to be safe) of a 1-2 year standby time with well over a month of talk time.

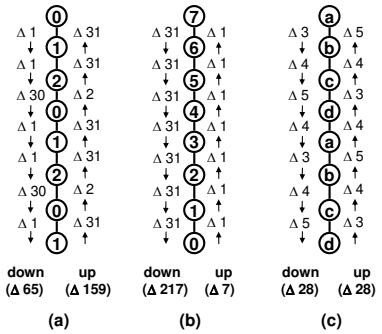


Figure 9. Different schedules change latency based on direction of the flow

Operation	Power	Time	Energy
4 bit ADPCM	$21mW$	$43\mu s$	$903nJ$
2 bit ADPCM	$21mW$	$37\mu s$	$777nJ$
ADC Sample	$21mW$	$3\mu s$	$6.3nJ$
RX Packet	$59.1mW$	$4ms$	$236\mu J$
TX Packet	$52.1mW$	$4ms$	$208\mu J$
Misc. CPU	$21mW$	$1ms$	$21\mu J$

Table 4. Energy consumption of voice streaming elements.

Battery	Sensing	Streaming
2 x AA	1.45 years	16 days
2 x D	(8.8) years	97 days
4 x D	(17.6) years	194 days

Table 5. System lifetime while idle and streaming. Values in brackets indicate calculated drain, however batteries typically have only a 5 year shelf life.

	TX Slots	RX Slots
a	0,8,16,24	3,11,19,27
b	3,11,19,27	7,15,23,31
c	7,15,23,31	4,12,20,28
d	4,12,20,28	0,8,16,24

Table 6. Expanded Voice Schedule Representation.

6.3. Coal Mine Deployment

We deployed a network of 42 nodes in the National Institute for Occupational Safety and Health (NIOSH)[27] experimental coal mine in Pennsylvania. The mine consists of over 2 miles of corridors cut out of a coal seam. Figure 10(b) shows an overhead map of the mine with our node locations. The walls inside a coal mine are typically residual coal pillars that the miners leave behind for support that allow almost no radio penetration. The dark lines show links that can be used for communication, while the dotted lines show weaker links that should be scheduled around, but avoid for communication. The AM transmitter would be connected at the mine entrance for global time synchronization along the backbone of the mine where a power line was available. Nodes placed in corridors away from the backbone used in-band time synchronization from nodes along the backbone. In Figure 10(a), each node in the graph is annotated with both the voice streaming and sensor delivery schedule. The voice slot assignment is abbreviated by schedules listed in Table 6. The numerical values represent the transmit slots used for communicating sensor data. The receive slot schedule (not shown in figure) corresponds to the transmit slots of the node's neighbors.

The network has two modes of operation. Under normal circumstances, sensor data are collected once every cycle (i.e. 6 seconds) from all nodes. This includes light, temperature, battery voltage and the SNR values associated with any nearby mobile nodes. During the audio streaming mode of operation, a mobile node was able to initiate a call to the gateway by the press of a button and stream bi-directional audio data. Our primary focus of this work was on the networking and evaluating the feasibility of such a system. For our tests, the mobile node was able to sample audio from the on-board microphone and compress the data while running the networking task. Our current mobile nodes do not have an on-board DAC and speaker output, so we used a computer connected to the node with a serial port to playback the received audio. To simplify tests, we transferred the encoded packet data over the UART and performed the decompression and playback live on the PC.

7. Conclusion

There is a growing demand for support of real-time streaming of voice over wireless sensor networks for emergency situations and intrusion detection. In this paper, we reviewed the design and deployment of the FireFly platform for real-time voice communication across multiple hops with timeliness properties, high throughput and predictable node lifetime. The hardware has a dual-radio architecture for data communication and hardware-based global time synchronization. To facilitate collision-free communication, we employ the RT-Link TDMA media access protocol. We implemented the ADPCM codec on our nodes and scheduled audio sampling, network and sensor reading tasks using the Nano-RK sensor RTOS. A 42-node network was deployed as an experimental rescue communication system in the NIOSH experimental coal mine. Our experiences demonstrate that:

- The FireFly platform was able to provide 2-way voice communication with a 24ms per-hop deterministic latency across 8 hops. The end-to-end delay was balanced in both directions.
- We were able to deliver robust call quality. Under low error rate conditions, it is better to use a higher quality codec such as 16Kbps ADPCM but under higher error rates, it is better to lower the stream quality to 8Kbps with redundant data.
- Support for real-time applications in sensor networks requires a tightly coupled cross-layer approach.

In the future, we would like to investigate link and path-disjoint redundancy for robust call quality. We are planning a new version of the FireFly node with a digital-to-analog converter to playback voice in mobile nodes. Provisioning real-time communication over sensor networks is an enabling technology for a new class of distributed applications.

Acknowledgments

The authors would like to thank Malolan Shantanakrishnan for his help with the final performance evaluation.

References

[1] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. *IPSN/SPOTS*, 2005.

[2] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *SenSys*, pages 214–226, 2004.

[3] N. Xu et. al. A wireless sensor network for structural monitoring. *SenSys*, 2004.

[4] D. Malan, T. Jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. *In Proceedings of the MobiSys Workshop on Applications of Mobile Embedded Systems*, 2004.

[5] Anand Eswaran, Anthony Rowe, and Raj Rajkumar. Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks. *IEEE Real-Time Systems Symposium*, 2005.

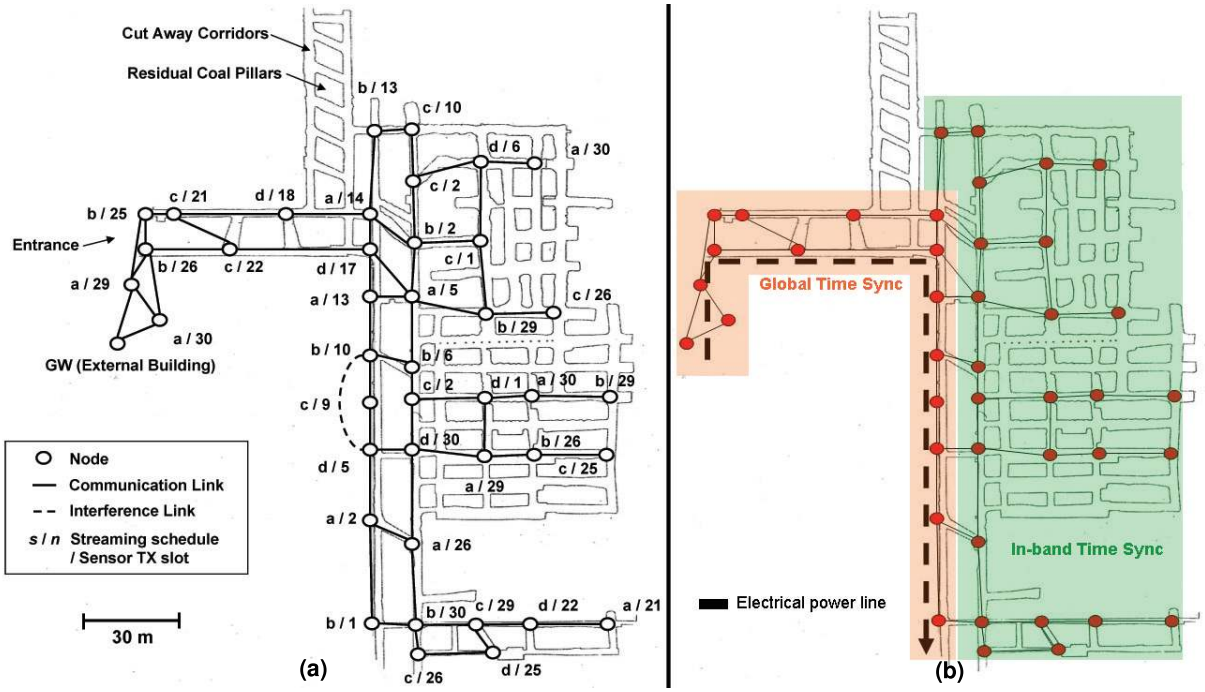


Figure 10. Schedule of voice and sensors collision free for NIOSH coal mine.

[6] A. Rowe, R. Mangharam, and R. Rajkumar. RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks. *Third IEEE International Conference on Sensors, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, Sept. 2006.

[7] T. F. Abdelzaher, S. Prabh, and R. Kiran. On real-time capacity limits of multihop wireless sensor networks. *RTSS*, pages 359–370, 2004.

[8] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. *RTAS*, 2002.

[9] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. *ICDCS*, 2003.

[10] T. Facchinetti, L. Almeida, G. C. Buttazzo, and C. Marchini. Real-time resource reservation protocol for wireless mobile ad hoc networks. *RTSS*, pages 382–391, 2004.

[11] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Sensys*, 2003.

[12] L.F.W. van Hoesel and P.J.M. Havinga. A lightweight medium access protocol for wireless sensor networks. *INSS*, 2004.

[13] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology management for sensor networks: Exploiting latency and density. *MobiHoc*, 2002.

[14] C. Guo, L. C. Zhong, and J. Rabaey. Low power distributed mac for ad hoc sensor radio networks. *Globecom*, 2001.

[15] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcast. *USENIX OSDI*, 2002.

[16] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. *Proc. ACM Sensys*, 2003.

[17] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. *Proc. ACM Sensys*, 2004.

[18] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. *Proc. ACM Sensys*, 2003.

[19] Adpcm using picmicro microcontrollers, app. note an643. microchip technology inc., 1997.

[20] Adpcm decoder on atmel, app. note avr336. atmel corporation, 2004.

[21] Atmel corporation, atmega32 data sheet, 2005.

[22] Chipcon inc., chipcon cc2420 data sheet, 2003.

[23] Knowles acoustics sp0103nc3 microphone data sheet, 2005.

[24] Radio systems 30w tr-6000 am transmitter data sheet, 2001.

[25] R. Mangharam and R. Rajkumar. Max: A maximal transmission concurrency mac for wireless networks with regular structure. *IEEE Broadnets*, 2006.

[26] C. Perkins, O. Hodson, and V. Hardman. A survey of packet-loss recovery for streaming audio. *IEEE Network*, 1998.

[27] R. J. Tuchman and R. F. Brinkley. A History of the Bureau of Mines Pittsburgh Research Center. *US Bureau of Mines*, pages 1–23, 1990.