6.267 Final Project

Colleen Josephson
cjoseph@mit.edu

2Lt Dave Adams, USAF
dcadams@mit.edu

Dec. 6, 2013

1 Introduction

This document is a design proposal for the link layer and media access control (MAC) mechanism of a seismic monitoring and warning network in California. This network consists of 100 monitoring stations, or sensors, distributed across the state, with one centralized base-station that all of the sensors report to. The network operates on the HF radio band, which enables communication over long distances due to ionosphere reflections. However, the propagation delay is quite high (20ms), as is the bit error rate (10^{-2}, after error correction). Additionally, there are occasional short drop outs lasting less than 20ms, as well as multi-hour stretches of bad channel conditions depending on ionosphere conditions. The network can operate up to 40 kbps, but if either a monitor or a base station have bad ionosphere conditions, then the rate is down to 4 kbps.

Under normal conditions, the sensors send hourly updates to the base station. Occasionally the base station will send out multi-megabit software updates. If an earthquake occurs, all the sensors that hear it need to send an emergency alert to the base station. The base station then selects a subset of the reporting stations to send a detailed report, which will contain measurements about the seismic activity that just occurred. The number selected will be about half of those that sent an emergency alert, and the earliest alerters are very likely to be selected for sending a detailed report.

The goals here are to create a system that maximizes efficiency under non-emergency situations, but to minimize latency while an earthquake is being reported. In other words, time-sensitive data is highly prioritized.

1.1 Assumptions

Throughout the design process, we worked with few important assumptions, which we believe are reasonable. These are outlined below.

- Seismic waves travel 4 to 8 km/s
- The sensors are located along fault lines, so that when a seismic wave reaches a fault line 10-20 sensors are activated at about the same time
- It is very fast (500μs) to switch between the 40 kbps and 4 kbps modes
- The base station has a much more powerful transmitter and receiver than the sensors, so it and the sensors can always hear each other, but the sensors cannot hear each other.
- The sensors have enough storage for a many-megabyte software update
- All of the sensors will be assigned a unique identifying number
- The base station has the storage and CPU capabilities of a powerful PC

- Sensors have GPS receivers, allowing for nanosecond-accurate time synchronization

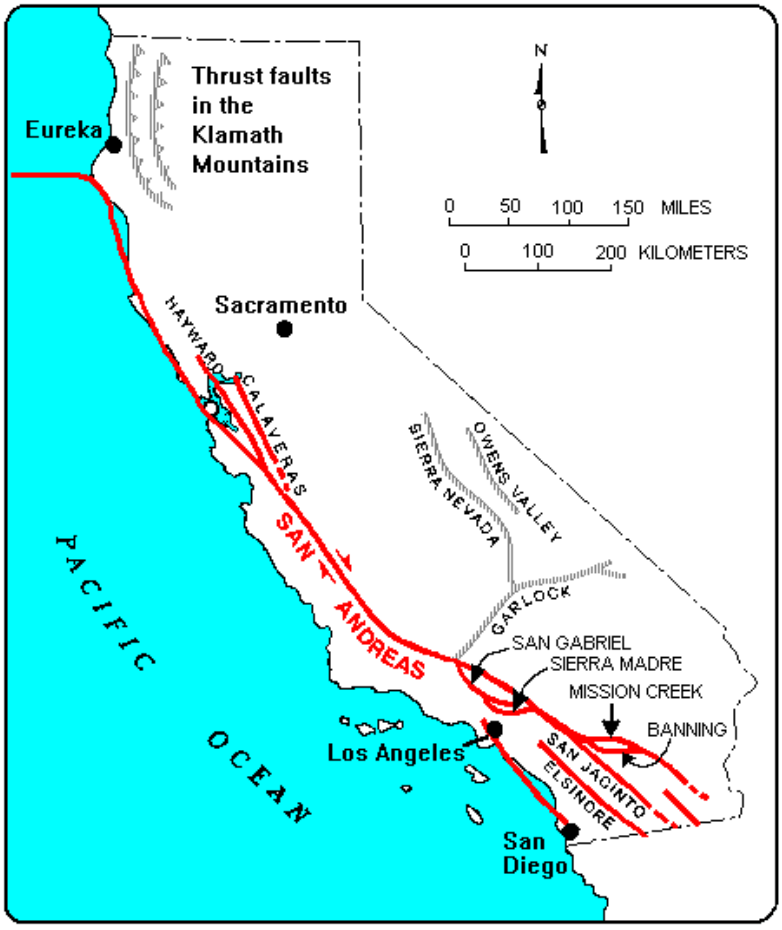

Figure 1: Image of California fault lines, from January/February 1992 issue of California Geology magazine

2 Design

This section outlines the design of the proposed protocol. There are three types of traffic the network must handle: emergency alerts, hourly reports, and software updates. **Table 1** gives details on these traffic classes. Due to the high bit error rate, these traffic types must be broken down into small packets and reassembled at the destination. To keep track of the packets, portions of the system will use a form of automatic repeat request, or ARQ. The link layer section will describe the packetization and acknowledgment system in detail. The MAC, or Media Access Control, section will describe how access to the channel is managed, and how to determine which sensors can send and receive at 40 kbps.

2.1 MAC

This section describes who can access the channel and at which times, which is known as a media access control, or MAC, scheme. The channel will be split into two modes of operation, normal mode and emergency mode. Each mode has its own TDM scheme, detailed below.

Traffic Class	Description	Size
Type I	Hourly reports from sensors to base-station, occasional commands from base-station to sensors. Can be delayed up to an hour.	10 kilobits
Type II	Emergency alerts from sensors when seismic activity over a certain threshold is detected.	500 bits
Type III	Software updates from base station to sensors. Occur infrequently.	Many megabits

Table 1: Descriptions of traffic types in the sensor network

.

2.1.1 Normal Mode

During the normal mode, the scheme will use time division multiplexing, or TDM, to split the channel into a 40 ms alert indicator period, a 199 ms broadcast period, and a 736 ms response period. Accounting for propagation delay and time needed to switch from 4 kbps to 40kbps, this adds up to a 1.036s TDM cycle. See Figure 2.

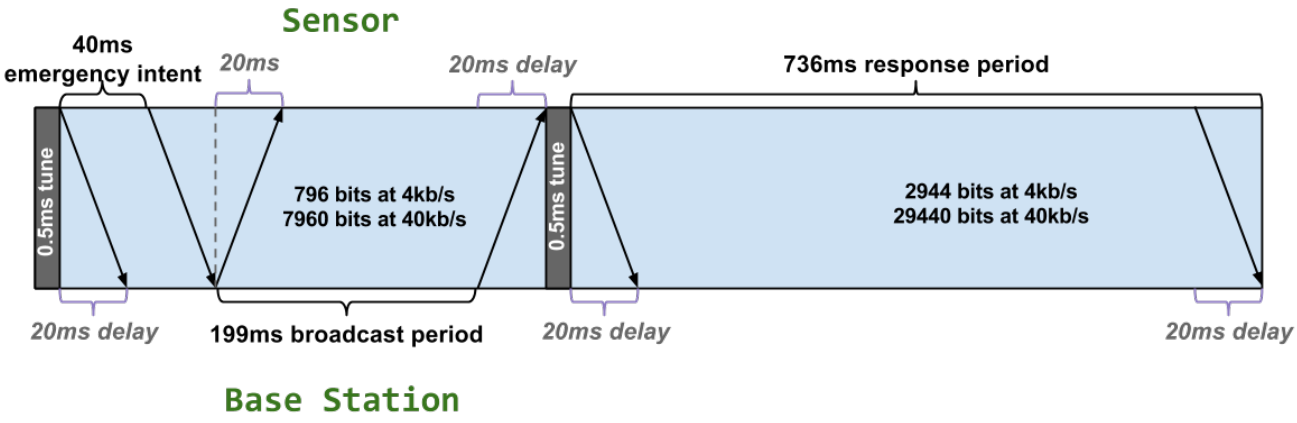

Figure 2: How the 1.036 second TDM cycle is divided during normal operation

The alert indicator period is random access. If a sensor detects seismic activity, it waits until the next indicator period and then broadcasts all 1's, indicating that it has an emergency alert to send. If enough sensors transmit during the indication period, the sensor will detect signal power above a certain threshold, causing the network to switch into emergency mode. This threshold technique means that false positives are a significant possibility depending on how low the threshold is. A single misbehaving sensor or random interference could trigger emergency mode. The threshold must be configured such that false negatives are unlikely, yet false positives are not too frequent. This period is timed so that it is longer than the occasional 20 ms dropouts, making it possible to switch modes even if a dropout occurs. In the event that an emergency alert is not triggered, the sensors will try again during the next indicator period.

During the broadcast period, the base station will acknowledge previously received packets and schedule transmissions to take place during the response period. It dictates which sensors should send hourly updates

and at which times and speeds. This also includes the possibility of reserving the channel for itself to disseminate software updates, opening the channel up to random access, or leaving the channel idle. The channel operates at 4 kbps during the broadcast period, as this rate works regardless of ionosphere conditions. With accurate GPS timing, it is easy for all quickly switch to receiving at the slow rate.

During the response period, the channel is used according to how the base station allocated it during the broadcast period. These uses include sending hourly reports, transmission of software updates, and sending probe packets during 40 kbps random access periods.

Probe packets are used to determine which sensors are capable of operating at 40 kbps. We are assuming that the ionosphere conditions are not universal across California, but we do not know the distribution of ionosphere states, e.g., if one sensor is in the bad state, how likely are other sensors nearby to be in the bad state? If the base station receives an intact message during the probe period, the base station updates the internal table it has to keep track of sensor bitrates. When the base station next wants data from that sensor, it will now request it at the 40 kbps rate. Degraded ionosphere conditions need no specific probe period. The base station can infer this by whether or not the data in a requested 40 kbps transmission is garbled. If all of the packets fail, then the base station will update that sensor's entry in the internal table and then re-request the transmission at the lower rate. If the base itself is in a bad state, it will look the same as all the sensors being in a bad state. Eventually the table will be updated to reflect the fact that the network must operate at 4 kbps. As the conditions improve, the occasional probe packets will bring it back up to speed.

Hourly reports are split into smaller packets and the sensor transmits as many as can fit in 736 ms. Depending on the rate and the packet header overhead, an entire data stream may need to be transmitted across multiple response periods. For this reason, the response time will be reserved entirely for one sensor. In order to successfully split a single transmission over multiple periods, a sensor needs to be able to hold state information about how fast the report should be transmitted and the highest sequence number transmitted so far, stop transmitting when the response period is up, listen for acknowledgments, and then resume transmission without prompting. First, the unacknowledged packets are re-transmitted, and then the sensor moves on to transmitting new packets. The sensor repeats this process until either the last sequence number is acknowledged, the base station sends a new request, or the base station changes into emergency mode. If the sensor finds that it still has left over time after transmitting what it needs to, it can start repeating unacknowledged packets to add redundancy.

Software updates are the lowest priority transmission, so these transmissions are only ever sent after the hourly reports have been successfully received. There will be no ARQ scheme for the software updates. Instead, we assume that the sensor can include their software version number in the hourly reports. The station will transmit the same software update repeatedly until it all of the hourly reports indicate the correct software version. Before transmitting the update, the base station needs to reserve the channel for itself. The base station will always transmit at 40 kbps, since any sensor that is not 40 kbps-capable will eventually be 40 kbps-capable during some future retransmissions of the update. If the sensor notices that the rate table contains no sensors capable of operating at 40 kbps, then it will wait to send the updates.[1] The sensors will keep any partially received software updates in storage until the update is completed. The sensors will also also only ever write the packet payload to disk if the packet was received correctly, meaning that the updates will only ever grow more complete over time. Because the packets will only ever be transmitted in order and there are no acknowledgments, the packets do not need sequence numbers. How does the receiver know when the software update is complete? Every time the update is transmitted, the base station will begin and end with start and end sentinel packets, and include the software version in the first packet. This way, a sensor knows that it is receiving the expected file. Once the base station begins broadcasting, it keeps broadcasting for many TDM cycles until the update is fully transmitted or the system is put into emergency mode.

[1] If no sensors can operate at 40 kbps this is a likley indicator that base station itself is in a bad state.

2.1.2 Emergency Mode

In emergency mode, all nodes switch to 4 kbps. The following 15 seconds divided into 20 equal slots.[2] (See Figure 3). Each sensor is assigned to a single slot. There will be 5 sensors per slot, so it is important that these sensors not be triggered at the same time with high probability. This can be done by choosing sensors on different fault lines, or geographically distant sensors on the same fault line. During its slot, the sensor broadcasts its 500 bit emergency alert. Each sensor will end up broadcasting its alert multiple times. This increases the probability that the alert will be correctly received after 15 s.

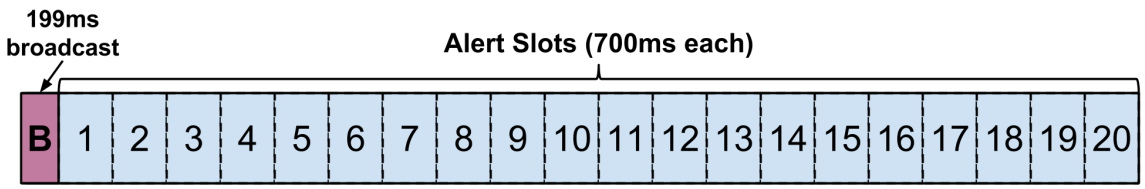

Figure 3: The 14.2 second emergency mode TDM

There are 20 slots because an earthquake typically triggers 10-20 sensors. If more sensors are triggered, there will be a collision, as per the pigeonhole principle. If fewer sensors are triggered, there will be wasted broadcast time. If two sensors happen to collide into the same slot, then neither's packets will get through. After the 20 slots are finished, the network changes back into normal mode so that the base station can collect detailed reports. The 15 second period is long enough for base station to receive all the necessary alerts with relatively high probability, with expected losses of only one alert out of twenty (see Analysis section for more detail). Thus if sensors have a slot collision or an emergency alert does not arrive successfully after the 15s period, there is no recovery mechanism. In making this design decision, we have assumed that the primary goal during emergency mode is early detection and warning, and that this task can be performed with acceptable accuracy even if there are some unaccounted-for alerts.

2.2 Link Layer

This section will get into the nuts and bolts of protocols used by the base station and sensors in each mode.

2.2.1 Normal Mode

Since the allocation of time in normal mode has been explained, we will now talk about the composition of the packets in each phase of normal mode. For the emergency beacon, sensors will transmit an all-one signal if they have detected seismic activity indicating an earthquake, and nothing otherwise.

The reservation phase will either be used by the sensor to disseminate commands or by the sensors to send hourly (or detailed emergency) reports. In the low-rate mode, this data consists of a frame of 128 packets comprised of an 8-bit SN, a 4-bit CRC, and 10 bits of data, as shown in Figure 4. Notice that the modulus of the SN is twice the frame size. The last frame of a report will be filled up with duplicate packets to further ensure receipt.

[2]The base station will be able to organize packets properly because of the highly structured nature of the TDM. All the packets in a slot are guaranteed to be from a single sender in a known order. The ordering allows for the base station to discard repeats of packets it has already successfully received. Once all of the packets that constitute an alert are received, the base station re-assembles the payloads and can extract the sensor ID from the alert metadata

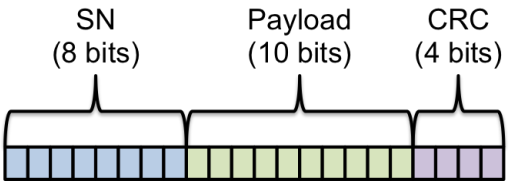

Figure 4: Structure of Hourly Report Packet
(Note that in high-rate mode SN is 10 bits)

At the high-rate mode, one frame consists of the entire 10 kb report, so the SNs are extended to 10 bits, while the structure of the packet is otherwise the same. Since there are only 1000 packets to be sent, the window doesn't slide, and a 10-bit SN suffices. Here, the priority in acknowledging packets is given to the longest sequences of successfully received packets, since there will certainly be more than 16 errors per frame.

For the broadcast phase, the base station will send packets consisting of a 7-bit ID number indicating the intended recipient, a 4-bit CRC, and 20 bits of data, as shown in Figure 5. If the base station is trying to broadcast in the next reservation slot, it will use an empty ID number with a specific payload. If the base station wants sensors to poll the high-rate channel, it uses an empty ID number with a different payload. Finally, to announce emergency mode, the base station uses a special ID number with maximal Hamming distance from all assigned ID numbers.

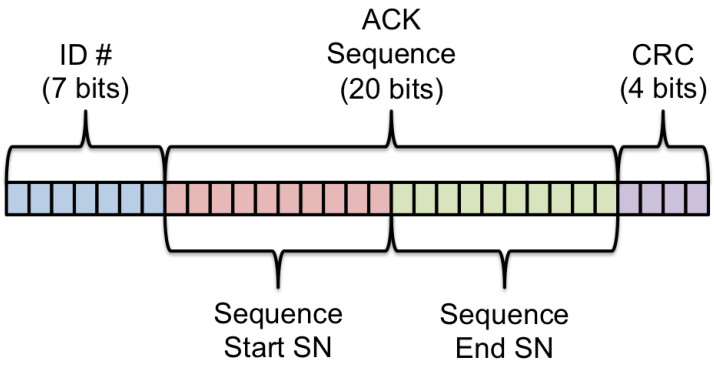

Figure 5: Structure of ACK Packet

When the base station is communicating with a specific node, the 20 bits of payload will communicate all possible messages. Most of the time, those bits will be used to present an ordered pair of sequence numbers indicating the start and end of a range of packets being ACKed. There can be 16 of these order pairs, each delivered 2 times to raise the reliability of delivery. To request an hourly report, the 10 bits representing the "start" SN will be all ones, while two distinct sequences for the "end" SN will indicate what rate the sending node should use. Since the 10 bits representing the beginning of a sequence would be higher than the 10 bits representing the end, the sensor would know that this means to send from the beginning.

2.2.2 Emergency Mode

The rationale of this design is explained in the Analysis section below, but emergency alerts are packaged as 10 bits of data with a 4-bit CRC. This block of 50 packets is sent 4 times in order without any sort of ARQ. Each sensor knows in advance its 700 ms slot.

3 Analysis

Now that the operation of the system has been completely explained, we need to look at how it performs.

3.1 Emergency Alerts

If we need to push through 20 emergency alerts in 15 seconds, then we should use all of that time and try to get the probability of error as low as possible. Since the system is designed to begin sending the emergency alerts within 1 second of the earthquake occurring, we can divide the remaining 14 seconds into 20 blocks of 700 ms. In each time block, we must push through 500 good bits. If we divide that data into K packets, each with a 4-bit CRC, then the length of each packet is $l = \lceil \frac{500}{K} \rceil + 4$, and we add redundancy by sending each packet N times.

Now then, the total number of bits sent for one alert is given by KNl. In 0.7 seconds at a rate of 4 kbps, we have 2800 bits to work with. We thus limit N, our amount of redundancy, by noting that

$$KNl \leq 2800$$

$$N \leq \frac{2800}{Kl}$$

$$N = \lfloor \frac{2800}{Kl} \rfloor$$

Since we have a bit error rate of 10^{-2}, then the probability of packet error is given by $p = 1 - 0.99^l$. We need only one of the N copies of the packet to be received for a successful reception of the packet, so

$$P\left[\text{packet not received}\right] = p^N$$

so that finally the probability of successfully receiving a given alert, or the probability of correctly receiving all packets comprising an alert at least one time, is given by

$$P\left[\text{successful alert receipt}\right] = \left(1 - P\left[\text{packet not received}\right]\right)^K$$

The resulting function is not well-behaved because of the various integer constraints, but we can optimize over K using and exhaustive search. Doing so shows a global optimum when $K = 50$, $N = 4$, and each packet contains 10 bits of data. Then probability of not receiving a complete alert is 0.01473 as shown in Figure 6. Consequently, the probability of receiving all 20 alerts in 15 seconds is

$$P\left[\text{system success}\right] = (1 - 0.01473)^{20} = 0.743$$

This is not great, but is acceptable. When we look at the expected number of correctly received alerts, however, the outlook is a bit better:

$$\lfloor n(1 - 0.01474) \rfloor = \lfloor 0.98n \rfloor$$

If twenty alerts were sent, we expect 19 of them to get through. This is pretty good, so long as it is acceptable to lose an alert or two.

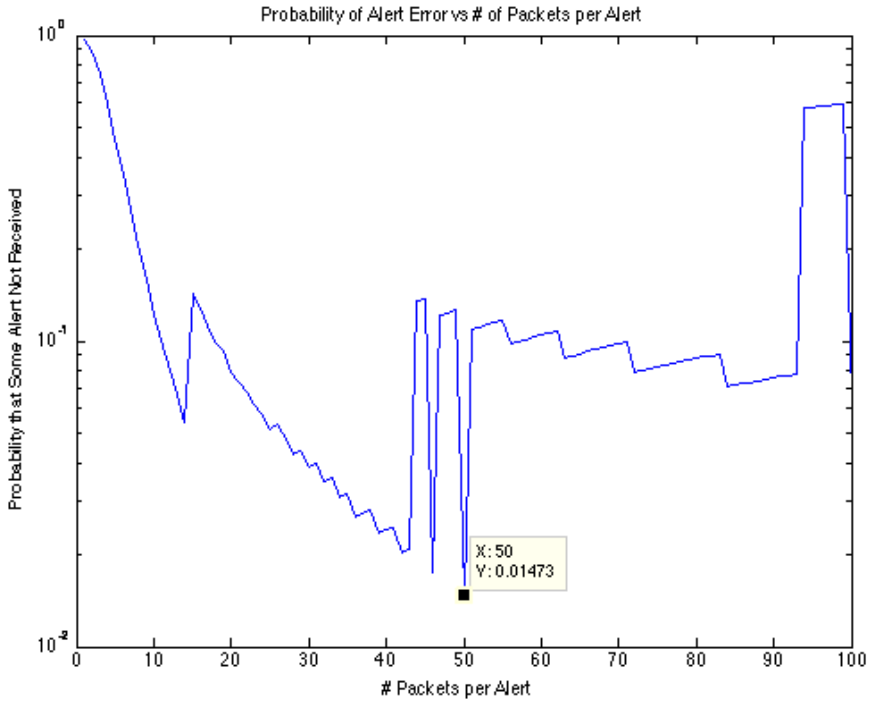

Figure 6: Probability of not receiving a given alert correctly

3.2 Hourly and Detailed Emergency Reports

The base station will request hourly updates from 40 kbps-capable sensors first, and then move onto the slower sensors. This takes advantage of the sensors in the good state while they still remain there, and gives the slow sensors a chance to enter the good state. While it is unlikely that a sensor will suddenly see improved ionospheric conditions over the amount of time it takes to send hourly updates because the coherence period is many hours long, the send rate could be set as 4 kbps even while the channel is 40 kbps capable, due to lost probe packets, lost ACKs, or timeouts.

We will look in detail at two particular performance metrics of the normal operation of this protocol: delay and efficiency.

3.2.1 Delay

Let's look at the expected amount of time it takes to get the detailed/hourly reports at the low rate. At a length of 22 bits, probability of packet loss is $1 - 0.99^{23} = 0.198$. Then

$$\mathbb{E}\left[\frac{\text{successes}}{\text{frame}}\right] = 128(1 - .198) = 102.6$$

and at 10 bits per packet and 1 frame per second, that means we should get the 10 kb report through in a little under 10 seconds, and thus get all 10 hourly reports in to the base station in 100 second, which just misses the 90 second goal.

As an upper bound, let us examine the performance if all packets can be transmitted at 40 kbps. At the higher rate, our packets are longer (24 bits), producing a packet loss rate of 0.214. Since we are acknowledging sequences of packets, it is no longer easy to analyze our throughput because we are limited by how many packets we can acknowledge. Throughput would accelerate after the first round, however,

because the ranges of packets grow, and unacknowledged packets are send multiple times. As an upper bound, however, consider a scenario where all packets could be ACKed. Then we our average number of packets per frame is

$$\mathbb{E}\left[\frac{\text{successes}}{\text{frame}}\right] = 1024(1 - .214) = 805$$

The remaining 219 packets would be sent 4 times each the next frame, resulting in a non-negligible chance at completing the transmission in two frames (over 0.5) and a high probability of completing the transmission in 3 frames. That means that transmitting ten hourly reports likely take less than 30 seconds, falling well within the 90 second goal.

During normal operation, we would expect to complete all our hourly reports in normal operation in roughly 17 minutes if we can only achieve 4 kbps with all sensors, and much faster than that if some sensors are capable of transmission at 40 kbps.

We must also look at the probability of getting the ACKs through to the sensors without error. Since there's no ARQ for ACK packets, we use the same optimization strategy on packet composition that we employed to reliably deliver the emergency alerts. Recalling that the length of these packets is 31 bits and noting that each of the 16 packets is sent 2 times, we find that the probability of receiving all ACK packets correctly is

$$P\left[\text{successful ACK delivery}\right] = \left(1 - \left(1 - 0.99^{31}\right)^2\right)^{16} = 0.30$$

That's pretty bad, but the probability of each individual ACK getting through is 0.93. At the low rate, we could send 8 different ACKs 3 times instead and get raise that 0.35 to 0.85. If it were possible that a stronger signal on the feedback channel could reduce probability of bit error to 10^{-3}, then we would be able to raise this probability to 0.985 without changing the packet structure.

3.2.2 Efficiency

In normal operation, efficiency is fairly low; a bit error rate of 10^{-2} requires a lot of redundancy and error detection to counter. However, our goodput is actually pretty easy to calculate. When sending hourly reports, we send exactly 10 bits of data per packet, and we send 128 packets per 1 seconds. Since, as calculated above, approximately 101.6 packets are successfully received each round, we get about 1.016 kbps, about 25% of our raw data rate. Since we can transmit all hourly reports in under 20 minutes, which leaves a lot of time for sending updates and probing, or simply leaving the channel idle to save power.

3.3 Software Updates

Software updates ate multiple megabits. When each packet has a payload of 10 bits, then a single megabit contains a 100,000 packets. At 40 kbps, 29440 bits fit into the allocated transmission time in a TDM cycle. When we factor in overhead, this fits 1178 25 bit packets. A one megabit update will take up 83 TDM cycles. Each cycle takes about a second, so each megabyte in an update takes up only about a minute and a half, thus it is possible to re-send updates many times in an hour, while leaving plenty of space in between for probing. The approach of retransmitting the entire update multiple times is not very efficient, but it is simple to implement and will eventually converge to having all sensors updated. Some small optimizations can be made to slightly improve the throughput, such as not bothering to transmit an update when none of the outstanding sensors are capable of 40 kbps transmission. This would mainly be useful for saving power.

4 Scalability

Scalability is always an important consideration in any system. The high bit error rate makes it very difficult to be able to expand the number of nodes while still maintaining an acceptable probability of receiving all alerts within 15s, or receiving all hourly reports from critical sensors within 90s. This is because the average number of nodes triggered during an event will increase, which means either more alert slot

collisions, or adding more alert slots, thus further subdividing the 14s alert period. Decreasing the number of times a sensor can send its alert even by one drastically affects the probability of success. For example, if we only transmitted alerts 3 times instead of 4, the probability of receiving all alerts within 15s becomes 0.1093, quite a downgrade from 0.743.

One possible problem with expanding the number of nodes is the necessity of increasing the number of bits needed for ID numbers. The network is highly sensitive to the number of bits in a packet, since the BER is so abysmal. There are currently 128 ID numbers, 127 of which are usable. If we want to expand beyond 128 sensors, then we would have to add an additional bit to ID field in the packets, which would impact the success rates rather significantly.

In summary, the system will probably perform well enough after adding an additional 27 nodes, which is an approximately 25% increase. However, expansions beyond this will perform quite poorly because of the increase in bits per packet as well as further time dividing an already thin channel.

5 Edge Cases

5.1 GPS Failure

This proposal relies heavily on GPS for accurate timing. If a sensor loses GPS lock, a local CMOS battery could keep time while waiting for the signal to re-establish. If the outage is short, clock skew will not have much time to build, so the channel should be able to successfully send an hourly report. The periods it would overlap with are blank air used for tuning to different bitrates, if necessary. Emergency alerts are less flexible, since each slot has at least one neighbor and no buffer space, making collisions more likley.

The base station losing lock is significantly worse when in emergency mode. The ability to accurately tell apart the 20 alert slots is crucial, otherwise there will be "off by one" type errors from subdividing the bitstream badly.

5.2 Very Large Earthquakes

If a very large earthquake occurs, more than 20 sensors might be triggered to send an emergency alert, yet we only have 20 slots. This will lead to one or more irresolvable collisions. There is currently no way around this, but an extension of the 14s alert period to add more slots would help if this edge case starts occurring frequently. This would reduce the likelihood of correctly receiving all alerts within 15s.

The followup period of collecting critical hourly reports will be more robust, since it uses both 40 kbps and 4 kbps. The current worst case for sending ten reports is 100s if all stations need to send at 4 kbps. Depending on how ionosphere conditions are correlated across sensors, some may be able to send at 40 kbps while others send at 4 kbps. The sensors at the fast rate will take up fewer rounds to arrive successfully, on average, so we can probably fit more than 10 reports within 90s successfully. This is difficult to analyze, however, since we have no information on the probability distribution of ionosphere conditions across a set of sensors.

5.3 Sensor failure

In the case of a sensor simply going offline or the transmitter breaking, the base station keep track of whether of not a sensor ever successfully transmits an hourly alert. If one or more hourly reports are missing, it would be a simple matter of programming to send out an e-mail alert and have a repair person go to the sensor location.

If the sensor's GPS unit breaks or loses lock for a long period of time, this could be catastrophic to the network in general, especially if clock skew builds quickly. In this case, we suggest that the sensors are programmed with a routine to bring the sensor offline should the GPS unit ever lose lock for too long a period of time. If the sensors have enough computation power, they could simply enter a loop that periodically checks back for lock. If this is not feasible, then the sensor can simply shut down. The base station will

eventually realize this, and send out an e-mail alert to have the sensor repaired.

5.4 Start-up and Base Station Failure

The base station going offline will bring the whole network down. However, this error will be recoverable. The network will simply enter the bootstrapping/ start-up phase once the base station returns. Of course, if this bootstrapping phase is not automatic, this outage could be a significant problem. We leave the design of a bootstrapping procedure for a later time.

5.5 Security

Security was not a design consideration. This system is highly vulnerable to an adversary, as it will be simple to eavesdrop and reverse-engineer the packet structure. Preventing this will be a difficult and costly endeavor, as precautions like encryption or random network coding add significant overhead to this already shoestring network. It is questionable that anybody would find value in sabotaging an earthquake detection network, but we cannot dismiss the possibility. If the equipment were upgraded such that the error rate is improved, then security measures may become possible.

6 Conclusion

To summarize, we have presented a multiple-access scheme for an earthquake-monitoring sensor network. The particular challenges of such a scheme were the high bit-error rate and the large latency of transmission. Our design is focused on prioritizing the delivery of emergency alerts indicating unusual seismic activity from the sensors to the central base station. In particular, the goal of a 1-second interval between a seismic event and the beginning of delivery of emergency alerts has required a significant sacrifice of efficiency. Although inefficient, this system is able to deliver emergency alerts within 15 seconds and still effectively pass on other kinds of data without getting backed-up.