The Day the RTCM Took Us Back In Time…

Dave Maciorowski
WA1JHK
13 February 2021

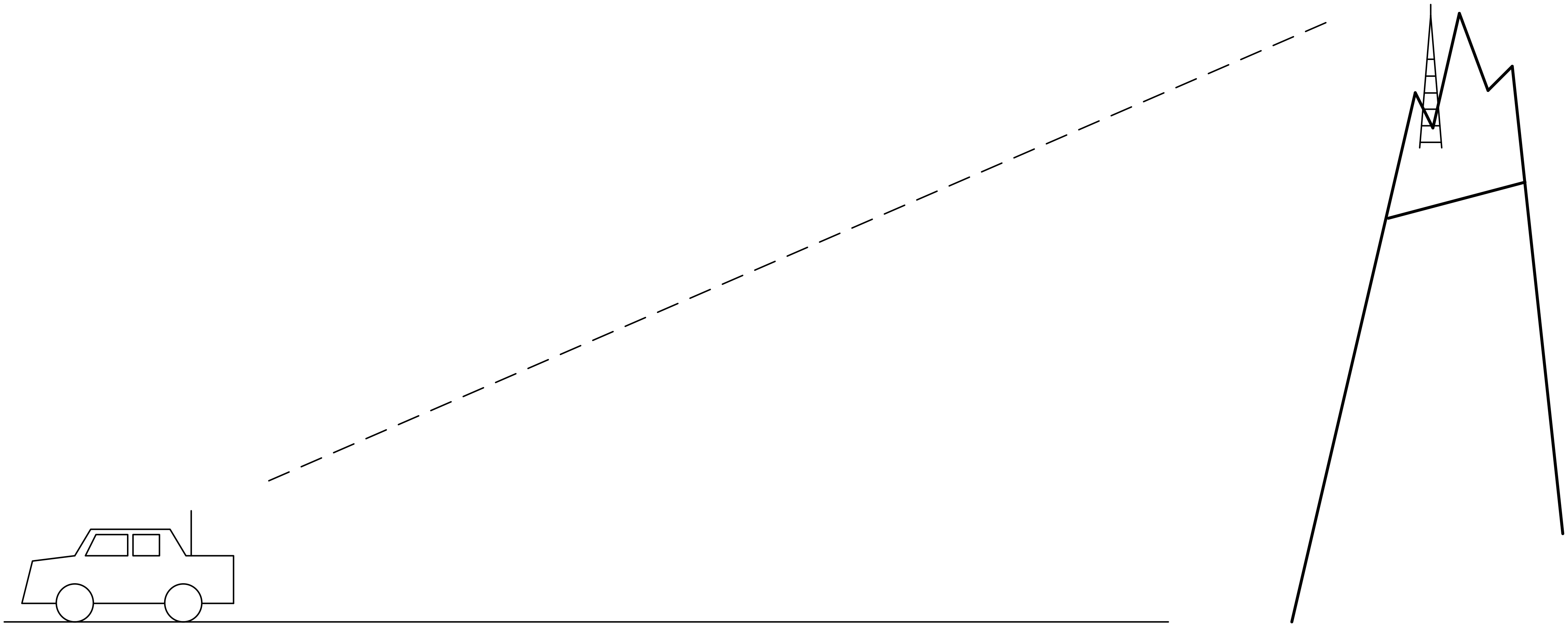
Abstract

- Y2K+21. Yes, it’s a thing.
- A look into time, how computers manage it and convert it, how the GPS system delivers it and how Radio Thin Client Modules (RTCMs) use it.
- Why a Y2K+21 bug took down all the RTCMs on 1/1/2021 at 00:00:00.
- Background of the problem and the fix...

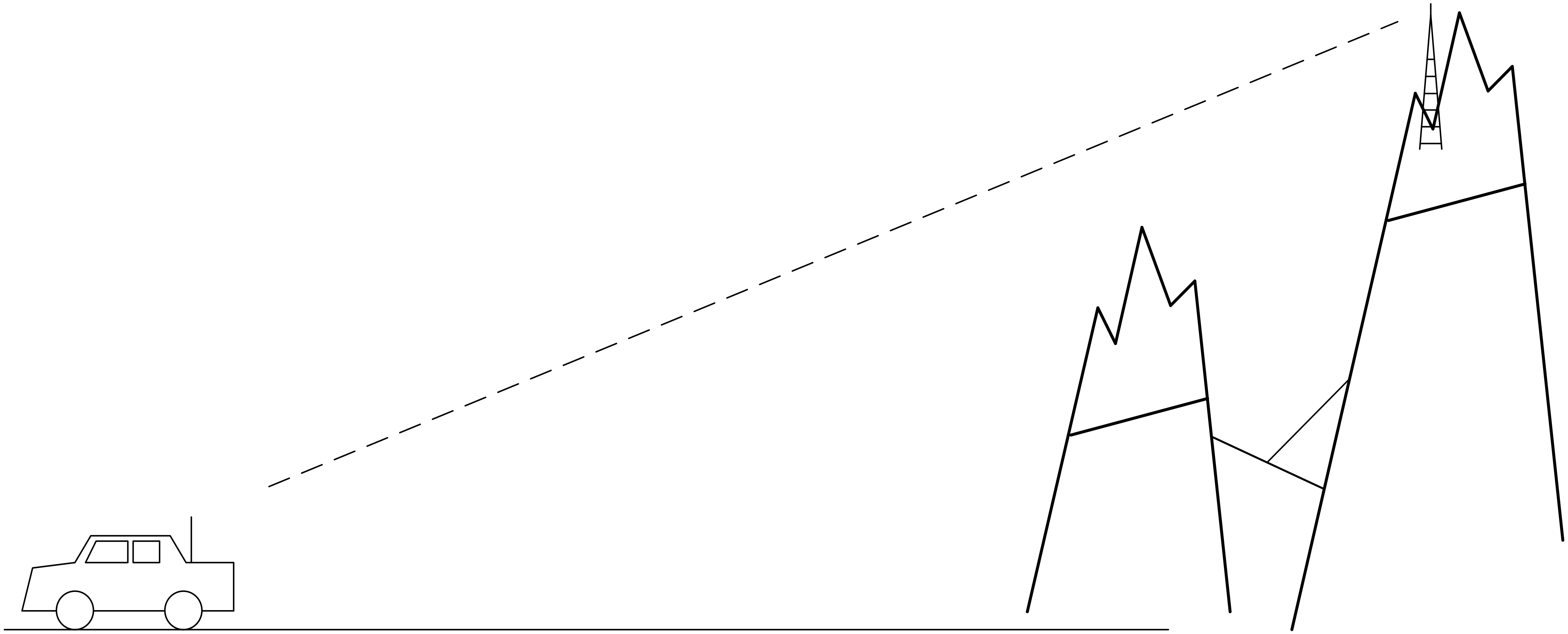
Overview

- Review of Receiver Voting
- Why is time important in Receiver Voting
- How the RTCM uses time
- Where the RTCM gets the time
- Y2K+21. OOPS
- The fix...

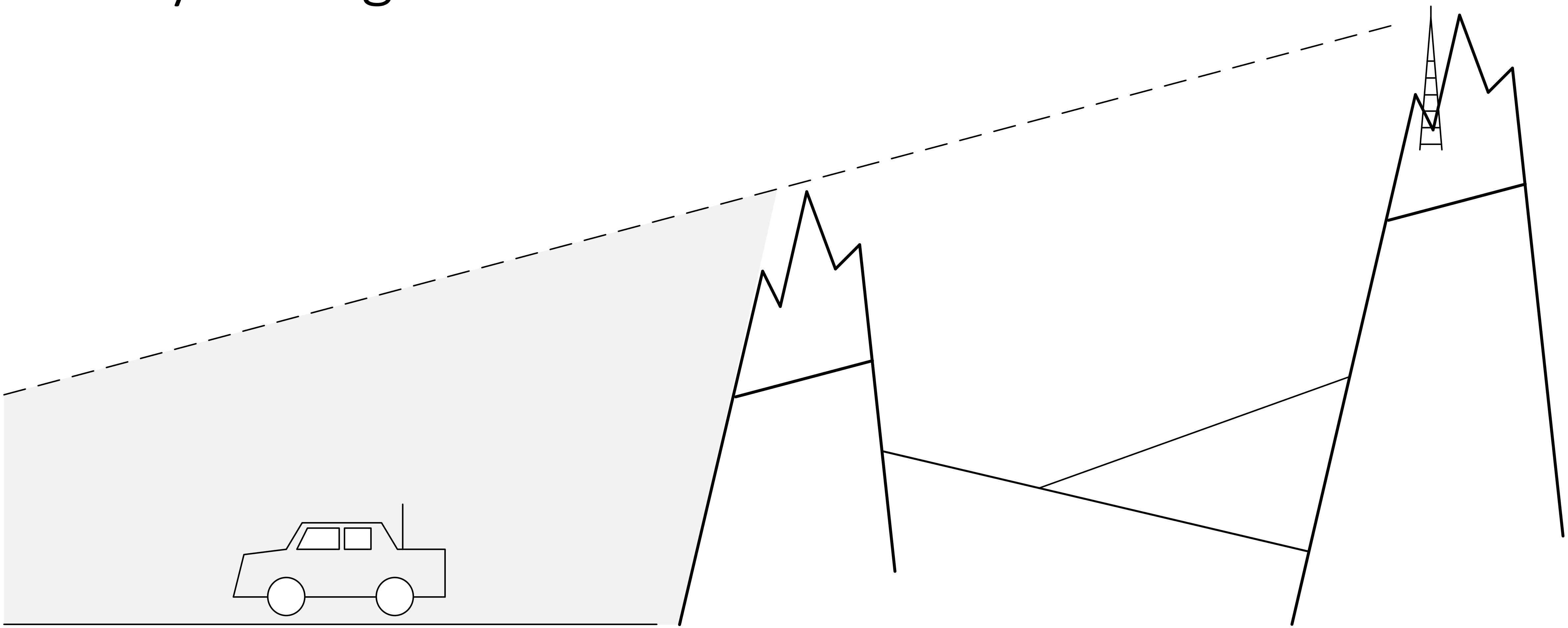
Why Voting? What's the Problem?



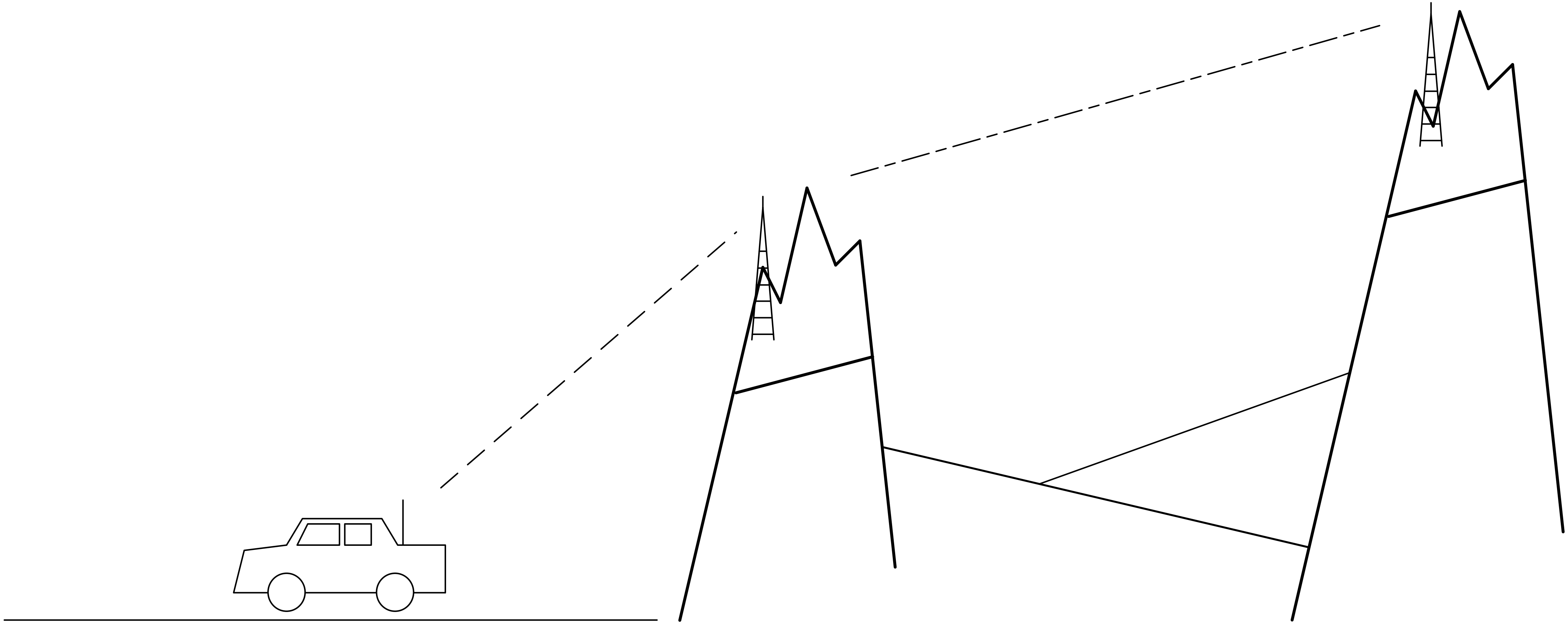
Why Voting? What's the Problem?



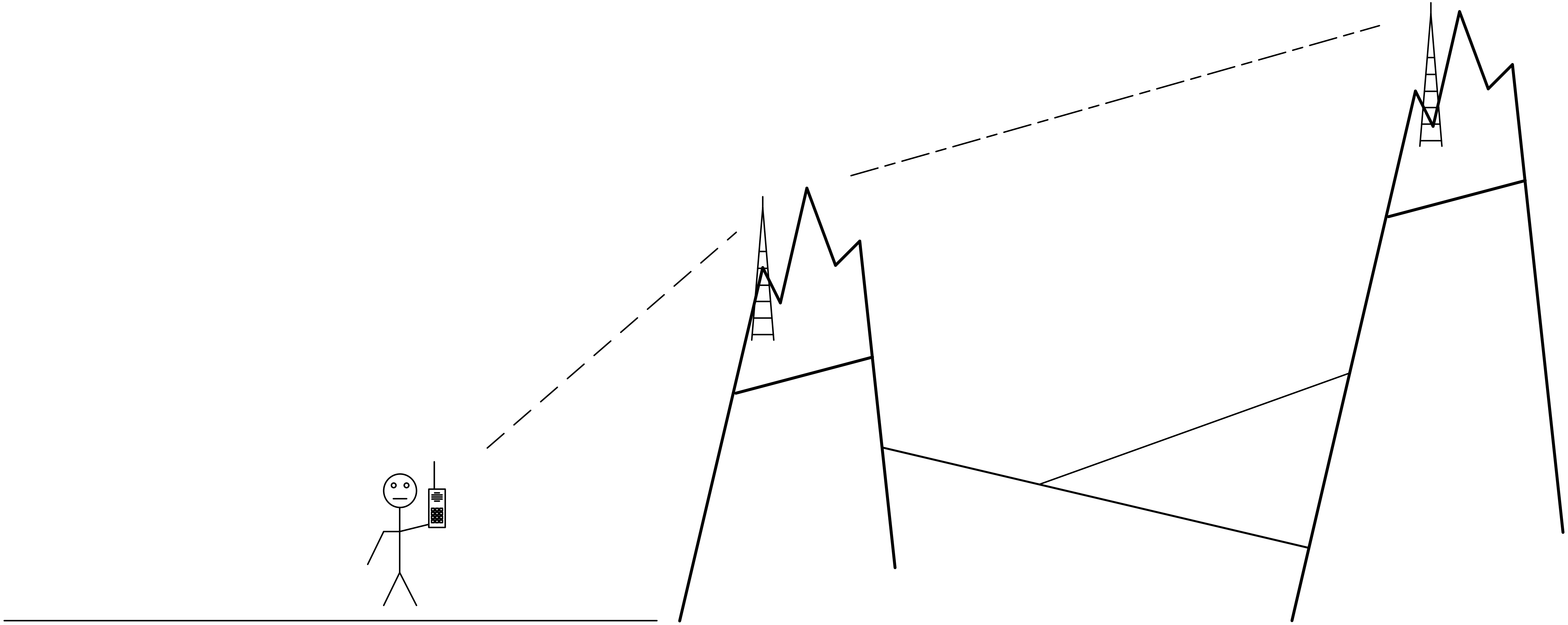
Why Voting? What's the Problem?



Why Voting? What's the Problem?



Why Voting? What's the Problem?

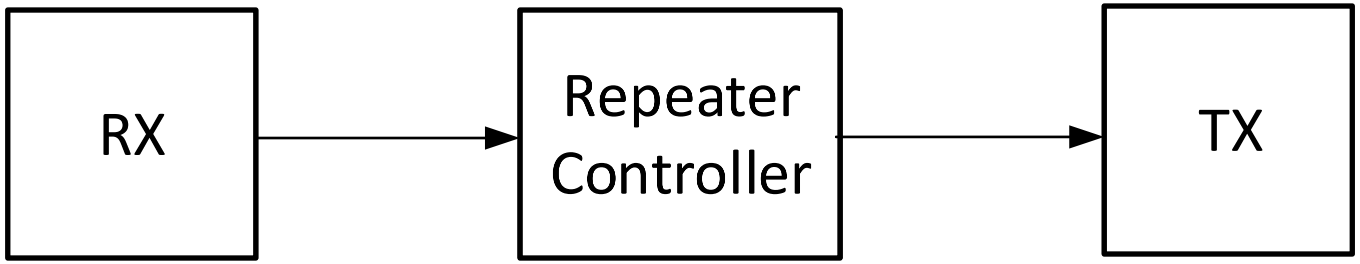


Why Voting?

- Benefits
  - Improved Coverage, especially for low power radios (handhelds)
  - Receiver Redundancy
- Challenges
  - Getting each receiver’s signal back to the Voter
  - Selecting the “best” quality version of the user’s signal
  - Delivering this selected signal to the transmitter
  - Complexity
  - Tuning!!! Getting the levels and audio quality right.

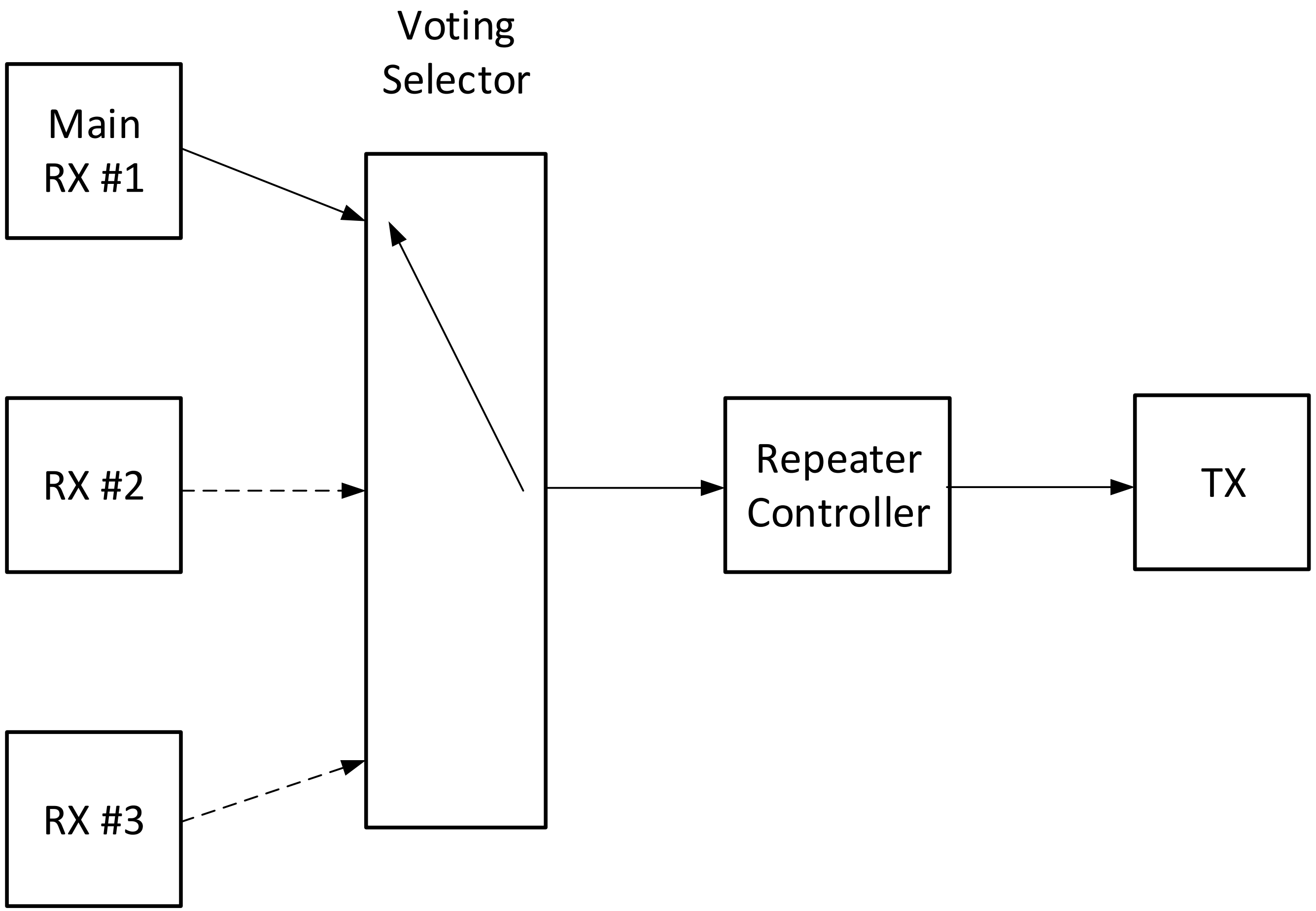
A Typical Repeater

- Repeater System with Main Receiver and Transmitter

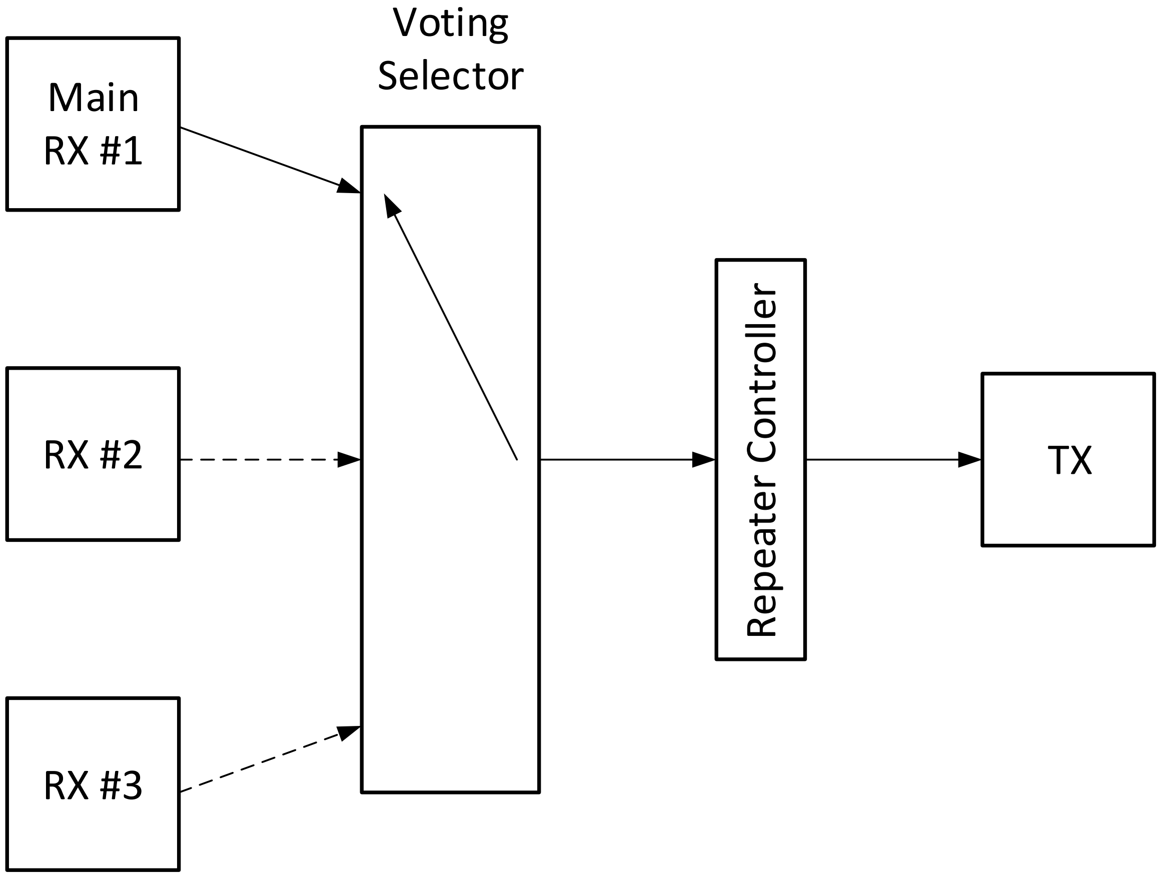


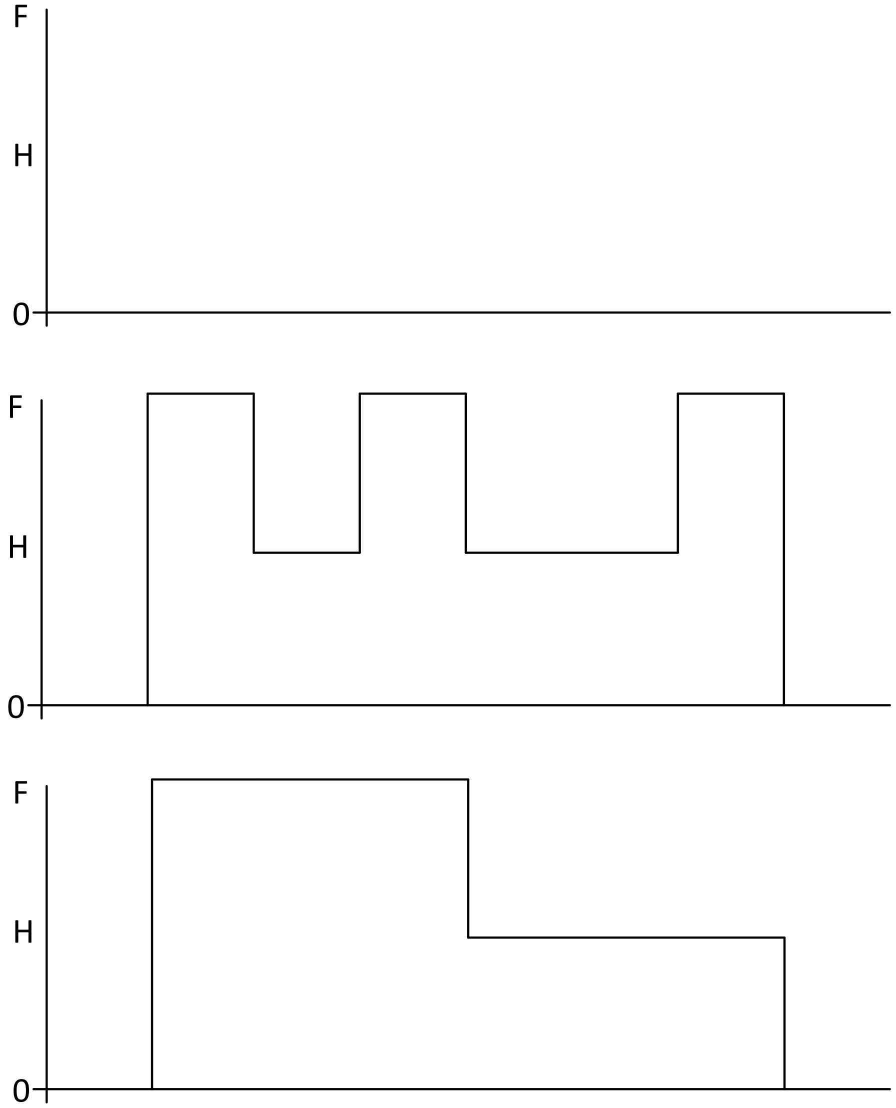
A Repeater with Voted Receivers

- Repeater System with Voted Receivers

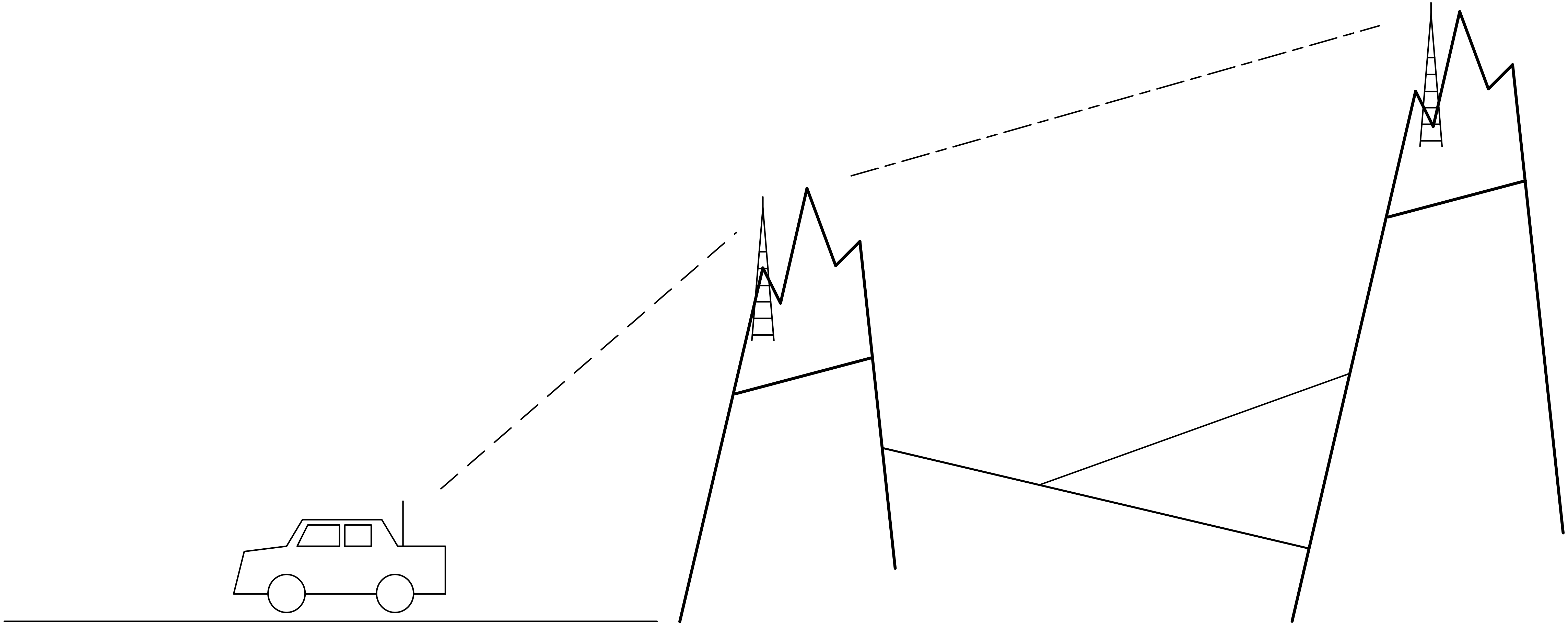


How Does It Work?



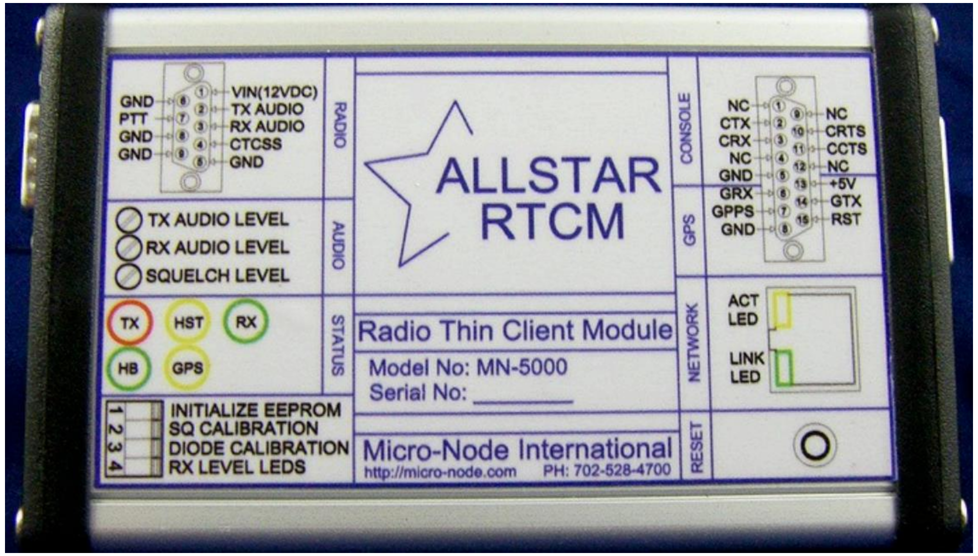


Getting Audio to the Main Site



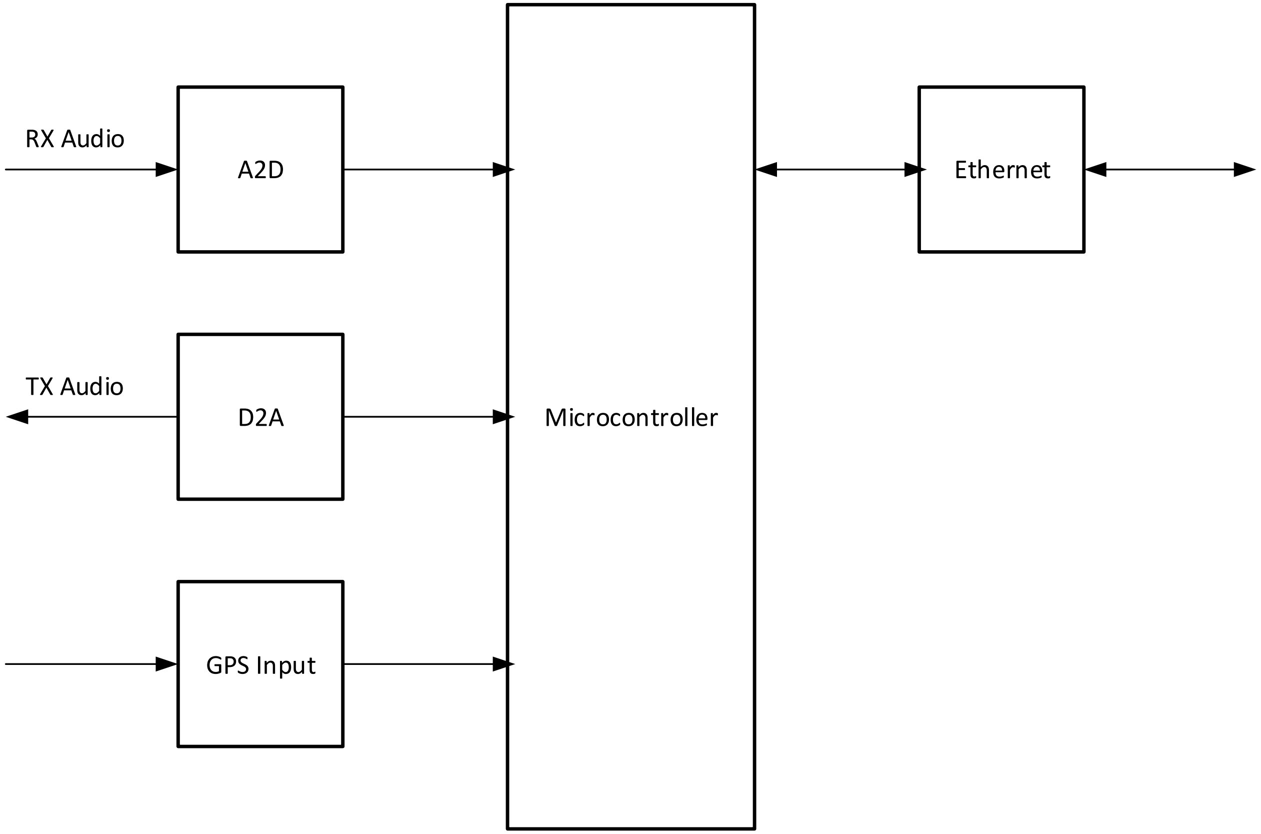
RTCMs

- VOTER
  - “Voice Observing Time Extension for Radio
- RTCM
  - Radio Thin Client Module
- Interfaces
  - Radio Interface
  - GPS Interface, Timestamps
  - Network Interface
  - VOTER Protocol
  - Setting Levels, Squelch
  - Fallback



RTCMs

- Device to get receiver audio signal back to main site
  - Digitizing RX Audio
  - RSSI
    - Received
  - Ethernet connection
    - Internet
    - IP over Microwave
    - Local Switch/Router
  - TX Audio in case we need it



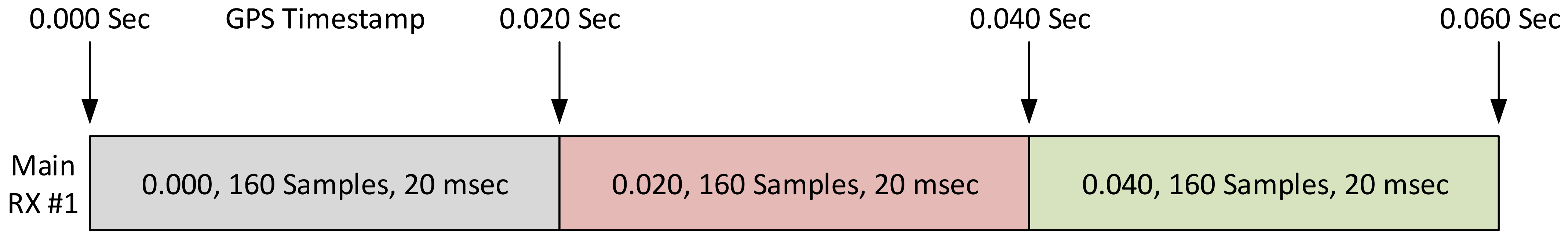
RTCMs

- VOTER Protocol
  - Transfers audio, timestamps and RSSI from receiver to voter selector
  - Sends node configuration information from the host to the receiver

- Packet Types
  - GPS Timestamps
  - Keep-alive
  - Receive Audio
  - Transmit Audio
  - Host Configuration
  - https://github.com/AllStarLink/voter

Moving the Audio

- Receiver Audio Over Ethernet



0.000 Sec

GPS Timestamp

0.020 Sec

0.040 Sec

0.060 Sec

Main RX #1

0.000, 160 Samples, 20 msec

0.020, 160 Samples, 20 msec

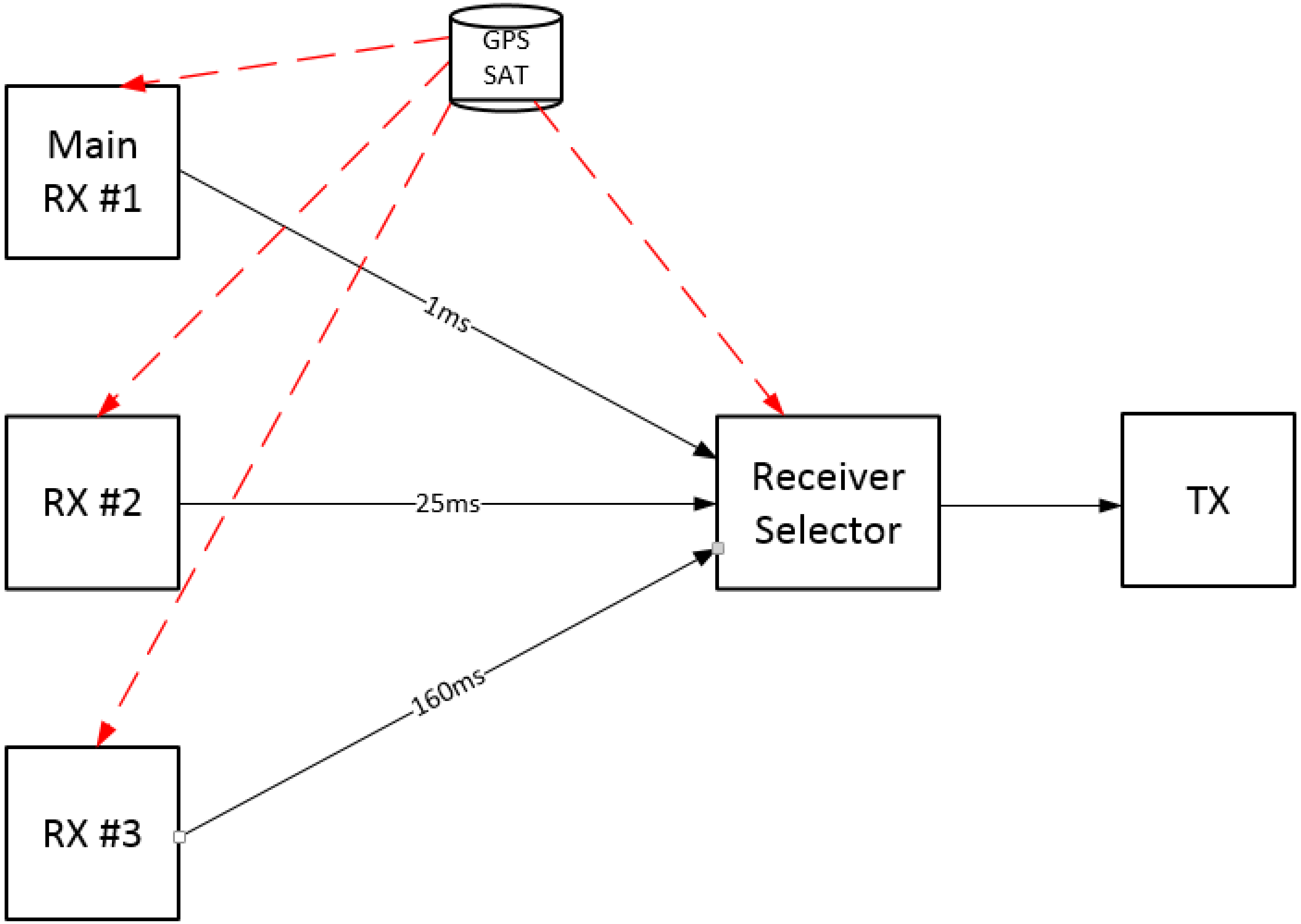
0.040, 160 Samples, 20 msec

Moving the Audio

- VOTER Protocol – Receive Audio Packet

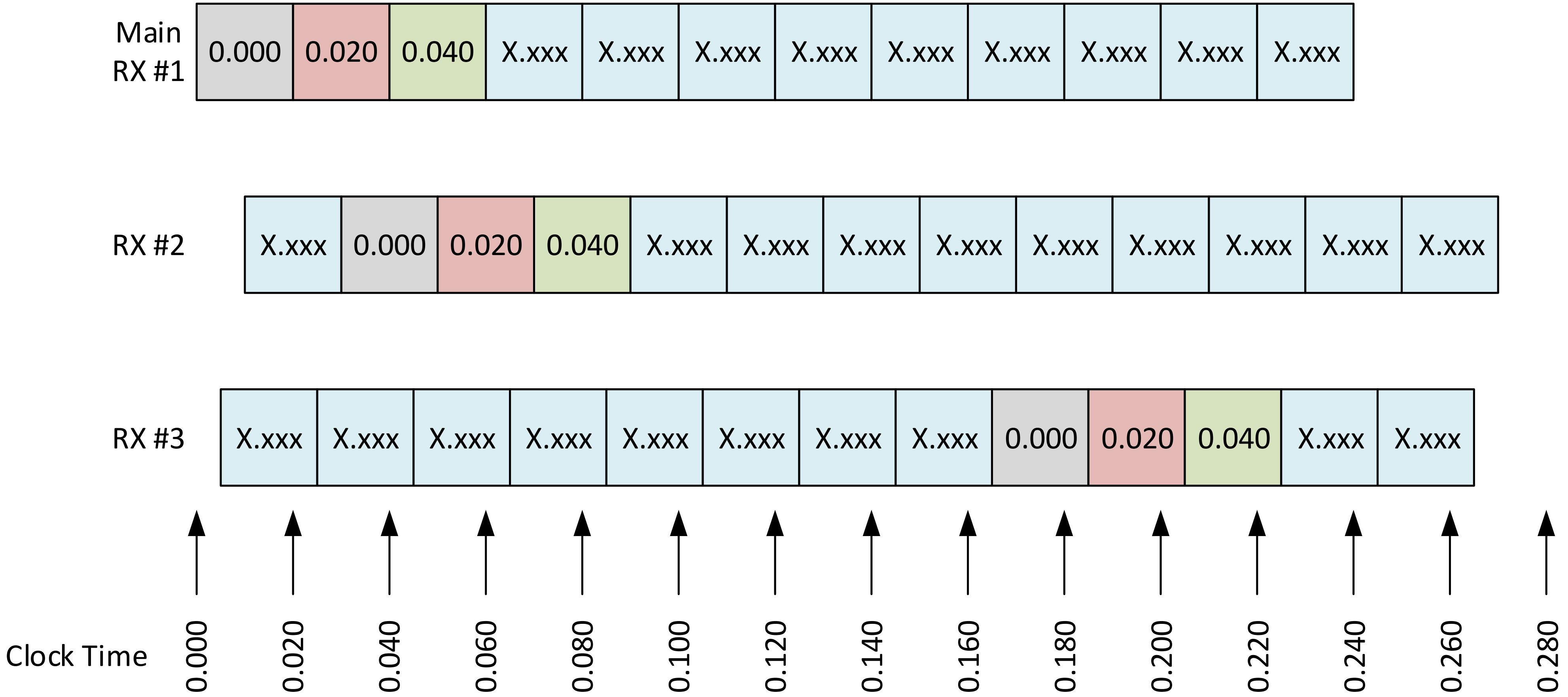
```
00:00:00.021761 IP 10.30.22.225.667 > 10.30.22.240.667: UDP, length 185
  Packet Type          : 1, RSSI Plus uLaw Audio
  Time Seconds/SeqNum: 1562862567.4875496
  Challenge/Response : 374725226/0xCAA2B26A
  RSSI (0xFF)          : 255
  Audio Samples, Length 160
    0x00: 5f5a58575a5d5b5b5e5f606262605e5c
    0x10: 5a59585d646671f1e7ea7779fb5f5757
    0x20: 564f4e4e504c4d52575a67e8e3ddd8db
    0x30: dae4fb696960595252514f4f5152555c
    0x40: 5c5b6aff6d6879f3776c6a756b686768
    0x50: 66645f5d595c5e5d5b5c63625f606162
    0x60: 6061666768 6c675f63625a57575d5b4e
    0x70: 4fe3e9454fdaf775674d57ec624e62e5
    0x80: eef9edf7ece9524856684e454d575556
    0x90: 5760fded797dded97966f5f55d5e6b64
```

Getting Time to the RTCMs



Moving the Audio

- Receiver Audio Over Ethernet, Variable Flight Time Latencies



Main RX #1 | 0.000 | 0.020 | 0.040 | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx

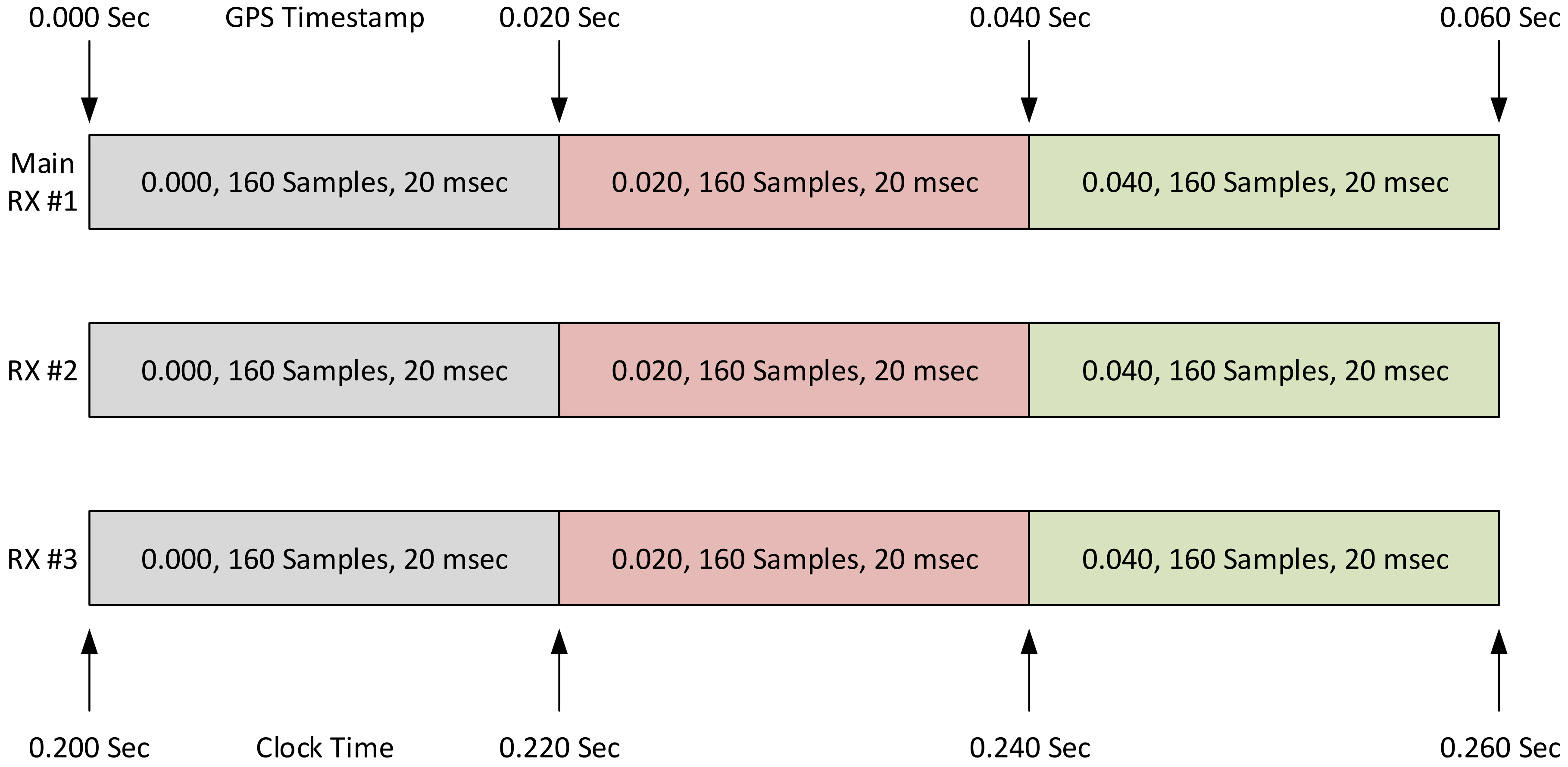
RX #2 | X.xxx | 0.000 | 0.020 | 0.040 | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx

RX #3 | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | X.xxx | 0.000 | 0.020 | 0.040 | X.xxx | X.xxx

Clock Time | 0.000 | 0.020 | 0.040 | 0.060 | 0.080 | 0.100 | 0.120 | 0.140 | 0.160 | 0.180 | 0.200 | 0.220 | 0.240 | 0.260 | 0.280

Moving the Audio

- Aligning Sample Buffers



0.000 Sec GPS Timestamp 0.020 Sec 0.040 Sec 0.060 Sec

Main RX #1	0.000, 160 Samples, 20 msec	0.020, 160 Samples, 20 msec	0.040, 160 Samples, 20 msec
RX #2	0.000, 160 Samples, 20 msec	0.020, 160 Samples, 20 msec	0.040, 160 Samples, 20 msec
RX #3	0.000, 160 Samples, 20 msec	0.020, 160 Samples, 20 msec	0.040, 160 Samples, 20 msec

0.200 Sec Clock Time 0.220 Sec 0.240 Sec 0.260 Sec

Y2K Again?

- Y2K happened because time was typically stored as two digits
  - “1999” was stored as “99”
  - The next year was stored as “00”, meaning “1900”
  - OOPS.
- C library time manages time as seconds since
  January 1, 1970 at 00:00:00 UTC
  - Is this a problem?
  - It will be for 32-bit integers at 03:14:07 UTC on 19 January 2038
- But why did it happen this year?
  - Y2K+21? Yes, it’s a thing.

The Beginning of Time

- Epoch
  - Computers use a fixed starting date time to calculate relative offsets
  - Different designers use different epochs
  - C Library internal time is managed as seconds since January 1, 1970 at 00:00:00 UTC
    - This relative time can be the cause of issues
  - How it’s stored and calculated can be an issue
    - Time limitations of 32-bit integers (signed value has only 31 significant bits)
    - Unsigned 32-bit DWORDS double the amount of available time
  - Thinking forward is sometimes a challenge for developers
    - Software lasts forever, but hardware ages and quietly fades away...

GPS Timestamps

- GPS receivers emit NMEA ASCII strings
  - Typical Time String
    - $GPRMC,194013.00,A,4032.94888,N,10511.83890,W,0.005,,020121,,,D*62
    - HHMMSS DDMMYY
  - The RTCM only requires the date and time to calculate seconds since epoch

Converting Time

- C Library time routines
  - mktime
    - Accepts individual time fields (time_t) as input
    - Outputs time in seconds since epoch
    - On error, it returns -1
      - -1 is an unusual time value and ignored in the code!

```
GPS-DEBUG: $GPRMC,201833,A,4004.3350,N,10521.2352,W,000.0,023.1,050121,008.8,E*6C
                  HHMMSS                                        DDMMYY
GPS-DEBUG: mon: 0, gps_time: -1, ctime: Thu Jan  1 00:00:0/ 1970
```

The Fix

```
#define SECONDS_EPOCH_TO_1121 1609459200  // seconds 1/1/1970 until 1/1/2021 0:0:0
#define SECONDS_PER_DAY 86400             // 60 * 60 * 24
#define SECONDS_PER_YEAR 31536000         // SECONDS_PER_DAY * 365

DWORD   total_seconds;

// days before current month in current year
static ROM int normal_year[] = {0,31,59,90,120,151,181,212,243,273,304,334};


// SECONDS_EPOCH_TO_1121 is seconds from 1/1/70 0:0:0 up to 1/1/21 0:0:0
total_seconds = SECONDS_EPOCH_TO_1121;
// seconds elapsed current day since midnight
total_seconds = total_seconds + ((DWORD)tm->tm_sec + ((DWORD)tm->tm_min * 60) + ((DWORD)tm->tm_hour * 3600));
// seconds elapsed since 1st of month up to current day
total_seconds = total_seconds + (((DWORD)tm->tm_mday - 1) * SECONDS_PER_DAY);
// seconds elapsed since 1st of year up to current month
total_seconds = total_seconds + (normal_year[tm->tm_mon - 1] * SECONDS_PER_DAY);
// seconds elapsed since 1st of year up to current month
total_seconds = total_seconds + ((tm->tm_year - 21) * SECONDS_PER_YEAR);
// seconds for leap day added for March thru December in leap year
if (((tm->tm_year % 4) == 0) & (tm->tm_mon > 2))
{
    total_seconds = total_seconds + SECONDS_PER_DAY;
}
// seconds for extra leap days for all past years
total_seconds = total_seconds + (((tm->tm_year - 21) / 4) * SECONDS_PER_DAY);
```

Fixed!

- Confirmation of the result

```
• GPS-DEBUG: $GPRMC,134238,A,4004.3361,N,10521.2338,W,000.0,322.5,080121,008.8,E*6D
•                   HHMMSS                                        DDMMYY
• GPS-DEBUG: mon: 1, gps_time: 1610113358, ctime: Fri Jan  8 13:42:38 2021
```

- Validating the Solution
  - Compare Against References available on the web
  - References
    - https://www.onlineconversion.com/days_between_advanced.htm
    - https://www.timeanddate.com/date/durationresult.html
      - This one had a leap year bug, now fixed

Questions?

References

- https://wiki.allstarlink.org/wiki/Main_Page
- https://wiki.allstarlink.org/wiki/RTCM_Client
- https://github.com/AllStarLink/voter

Backup

Calculation Considerations

- Is there enough time?
  - Microcontrollers have limited execution time available
    - The VOTER uses a 16-bit microcontroller
  - Application requirements can limit available time
    - A VOTER must collect and send 160 samples every 20 milliseconds

Overview – Some Definitions

- VOTER
  - The first hardware module
  - Network Protocol
  - VOTER = “Voice Observing Time Extension for Radio”
- RTCM
  - RTCM = Radio Thin Client Module, a device previously manufactured by Micro-Node International
- RoIP
  - Radio Over IP, communicating audio and control signals (COR, PTT) across an Ethernet connection.
- RSSI
  - Received Signal Strength Indicator, how strong are you into that receiver?