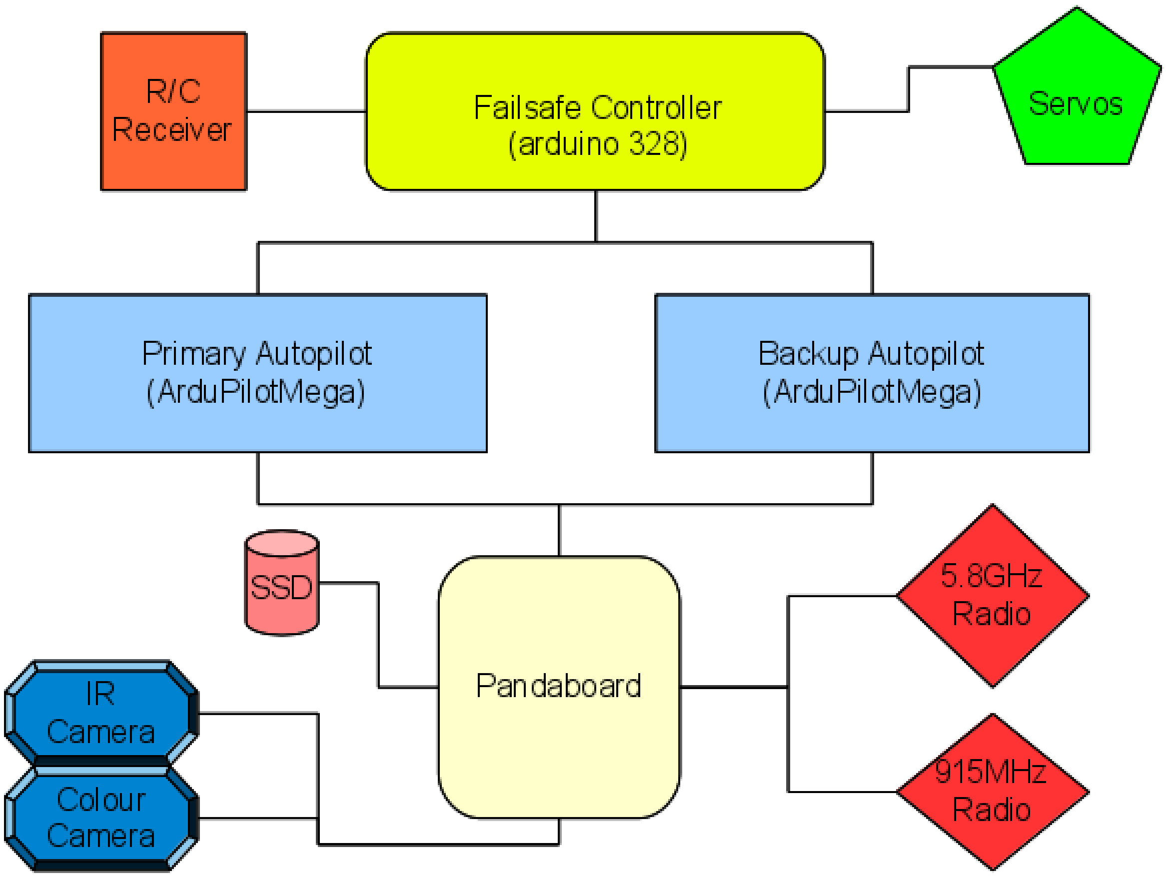
Building an Open Source Telemetry Radio

Andrew Tridgell
(VK1FAAH)

Communicating with UAVs

- Telemetry for UAV ↔ ground station comms
  - real-time flight data
  - reliable control and override
  - low bandwidth (typically 3 kbytes/second)
  - highly repetitive data (e.g. 4Hz sensor data)
- MAVLink protocol
  - Micro Aerial Vehicle communications protocol
  - encapsulates sensor, fight data and control data
  - XML based protocol definition

UAV Schematic



3DR Telemetry Radio

- Si1000 SoC
  - 8051 embedded micro, 25 MHz
  - RAM: 128 + 256 + 4096
  - 64kbyte EPROM for firmware
  - GFSK modulation
  - TTL serial interface
  - 433 and 915 MHz variants (868 and 470 possible)
  - 20dBm max transmit power
  - -121dBm receive sensitivity

Existing firmware

- Very simplistic existing firmware
  - copies bytes from serial to radio
  - copies bytes from radio to serial
  - no attempt at avoiding collisions
  - no frequency hopping, no LBT, no encapsulation
  - no attempt at complying with licensing rules

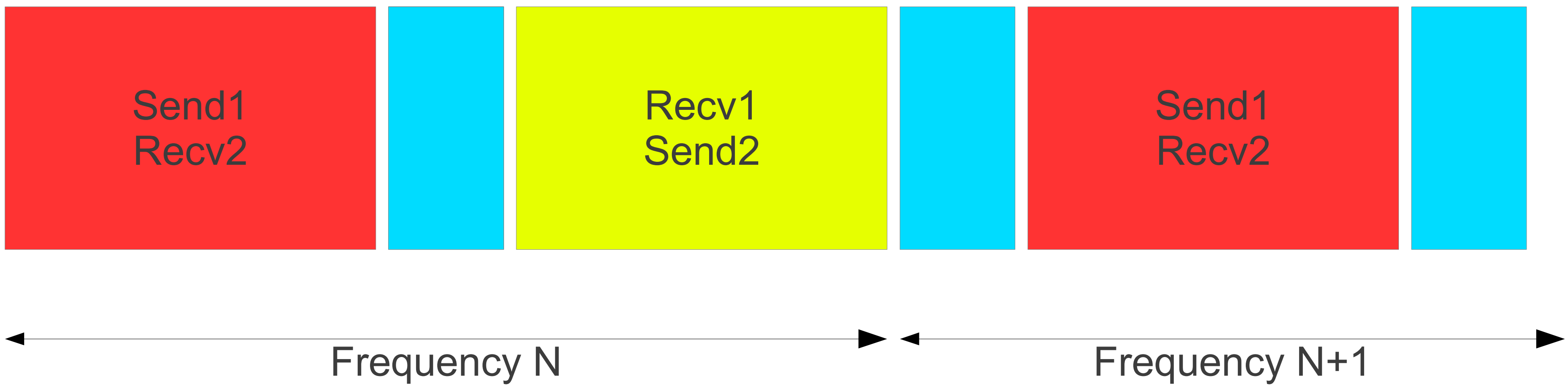
Coding for Si1000

- SDCC Compiler
  - specialist C compiler for small devices
  - 3 memory models (small, medium, large)
  - not stack based by default
  - need to tag variables with memory type
  - support for boolean single bit types
  - nice support for critical sections

Avoiding collisions

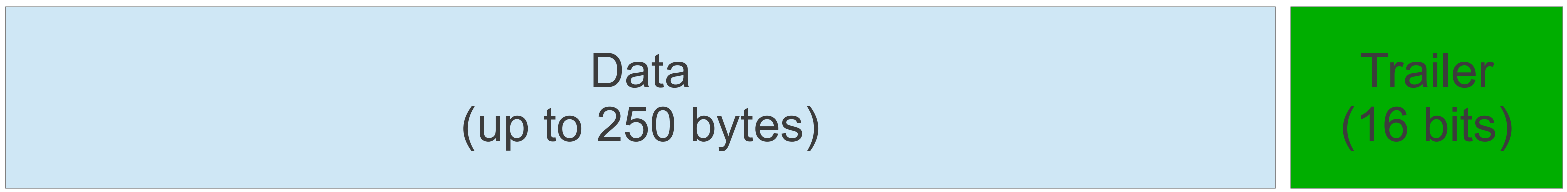
- Don't all talk at once!
  - radio can either be listening or sending, not both
  - only one frequency can be tuned at a time
- How to avoid collisions?
  - simplest solution is to have “time slots” for transmission for each radio
  - this is called TDM (time division multiplexing)

Time Division Multiplexing



Send window: typically 100ms (6200 ticks)
Silence period: typically 5ms (360 ticks)

Time Division Multiplexing ...




```
struct tdm_trailer {
        uint16_t window:13;
        uint16_t command:1;
        uint16_t bonus:1;
        uint16_t resend:1;
};
```

- Adaptive timing
  - sender gives up time slice with zero data send
  - 'window' is number of 16 usec ticks remaining
  - 'command' allows for remote command operations
  - 'resend' allows for opportunistic data resend
  - note that this all works for 1-way links too

Frequency Hopping

- Changing frequency regularly helps
  - allows more users of same frequency band
  - required for compliance in many countries
- Frequency hopping in Si1000
  - registers for base frequency and channel separation
  - register for current channel

Frequency Hopping ...

- Randomised channels
  - create random channel order based on network ID
  - switch channels at end of each TDM cycle
- Initial search
  - until lock achieved, move receive frequency slowly
  - lock is achieved by single received packet

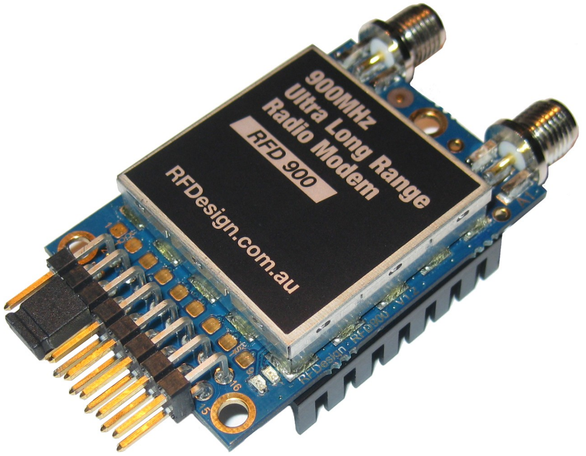
Error Correction

- Losing one bit can be bad
  - single bit error causes packet loss
  - how to handle errors?
- Error correcting code
  - many available, chose Golay 23/12 code
  - same as used by Voyager 1 & 2
  - table based in flash, very low memory use
  - corrects up to 3 bit errors per 12 bits of data
  - halves bandwidth, but increases noise robustness

Other features

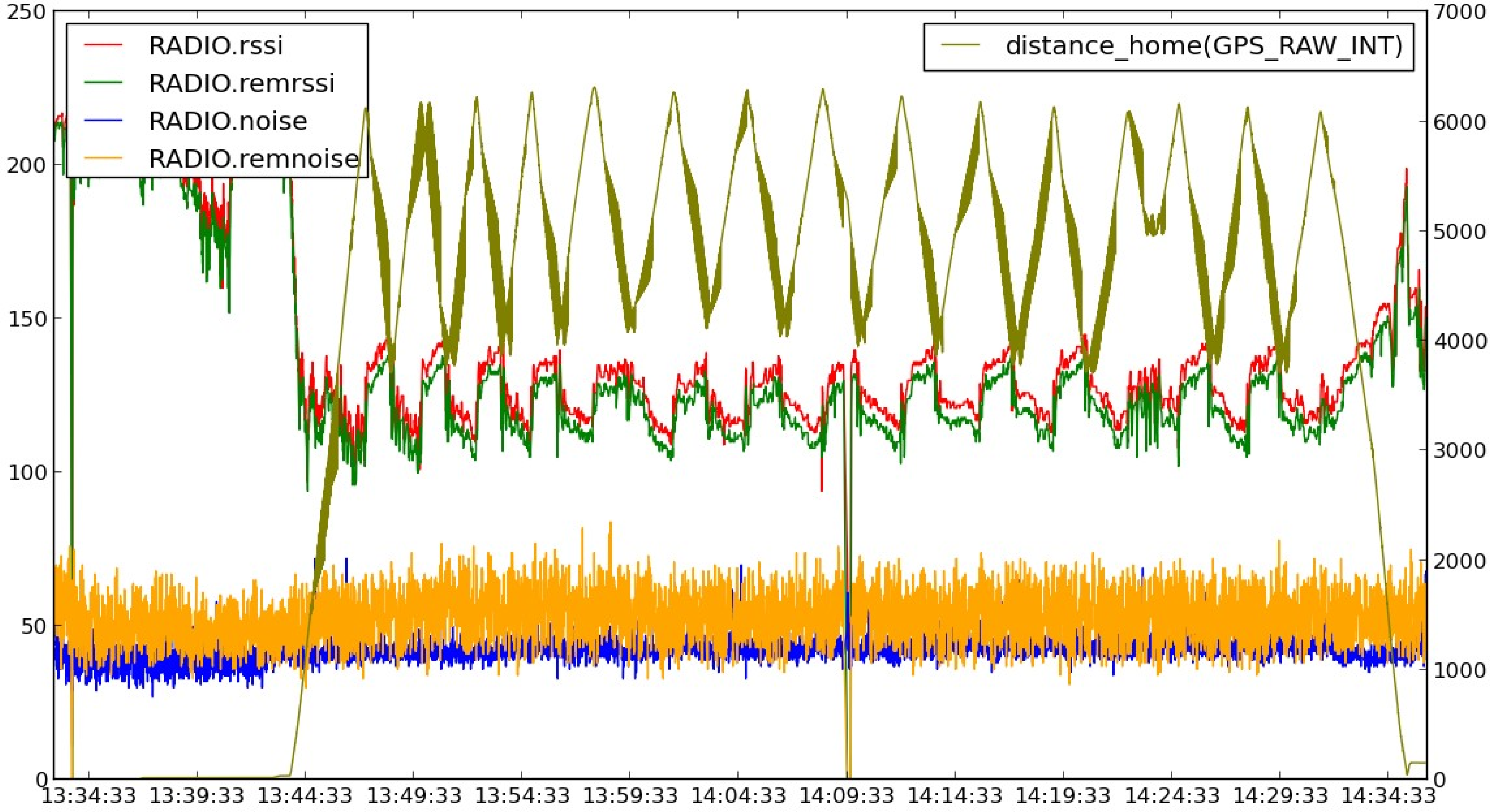
- Regulatory compliance
  - Listen Before Talk (LBT) for EU compliance
  - duty cycle for EU compliance
- User control
  - 'AT' interface for configuration
  - 'RT' interface for remote configuration
- MAVLink features
  - mavlink framing for lower data loss
  - 'RADIO' MAVLink packets for reporting and flow control

More range - RFD900



- Longer range needed for S&R UAVs
  - Collaboration with RFDesign in Brisbane
  - Added 20dB power amplifier (PA)
  - Added 20dB low noise receive amplifier (LNA)
  - Added RX SAW filter and TX low pass filter
  - Added antenna diversity
- Much more range
  - Range of around 60-80km with omni antennas
  - Only small firmware modifications required

Range testing

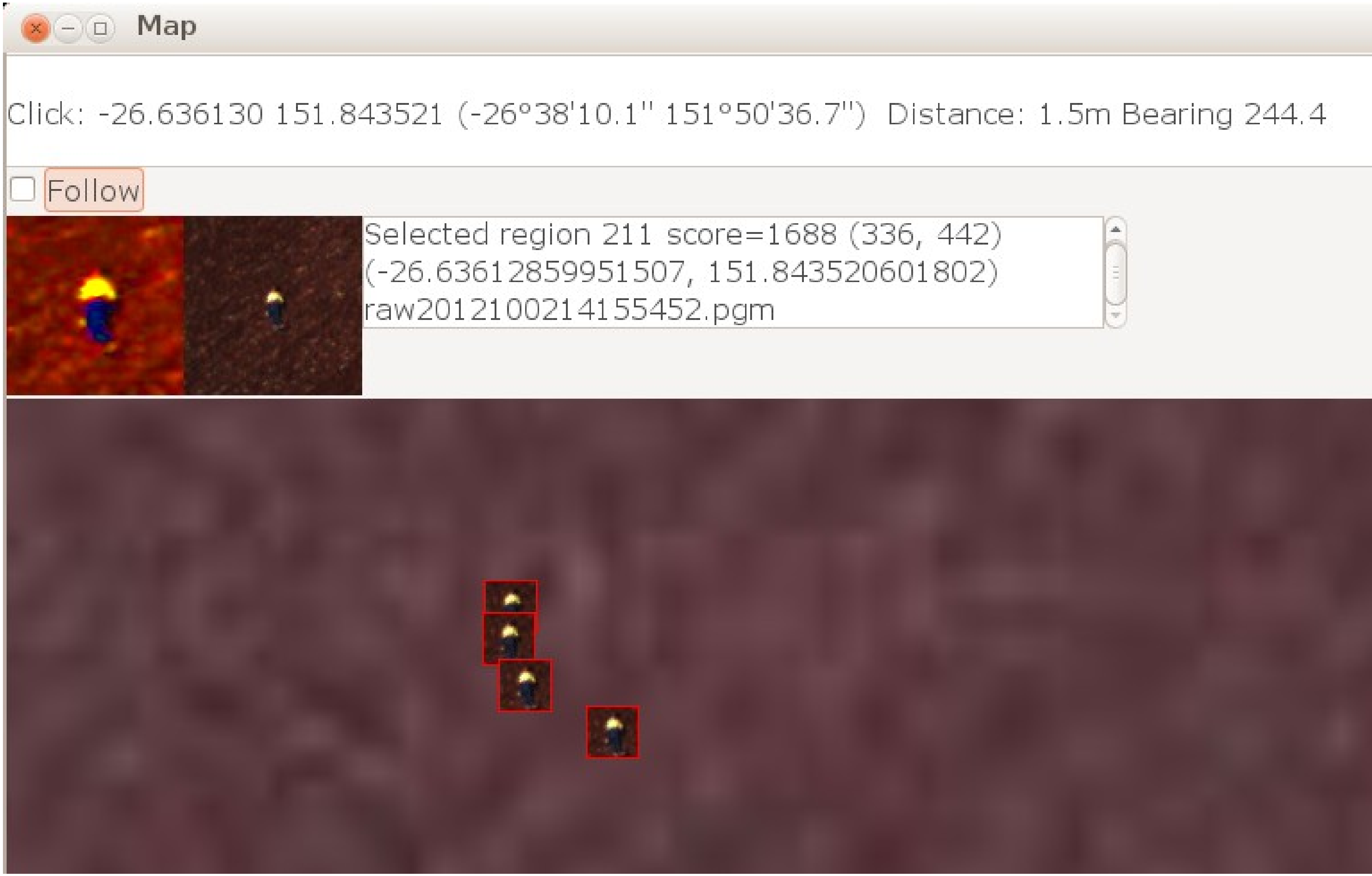


Noise Testing

Transferring images

- A telemetry radio is great for telemetry, but what about images?
  - typically much higher bandwidth requirements
  - non-repetitive data, usually not time critical
  - needs guaranteed delivery for S&R target images
- CanberraUAV setup
  - one RFD900 and one Ubiquity 5.8GHz bridge
  - full redundancy, mission completion with either radio

Full versus Thumbnail



The problem with TCP

- Initially tried TCP for image transfer
  - very poor handling of packet loss
  - largely assumes loss is congestion
  - changing congestion control algorithm didn't help
  - very poor control over bandwidth usage
- UAV communication is unusual
  - single user of radio link – greed is good!
  - link loss varies widely during flight, from 5% to 95%
  - need to use available bandwidth efficiently
  - at 90% packet loss, should get 10% throughput

BlockXmit Protocol

- New protocol for block data transfer
  - user specified bandwidth and segment size
  - user supplied packet encapsulation
  - greedy use of bandwidth. If you aren't sending, you are wasting bandwidth.
  - extent based acknowledgement system
  - multiple blocks in flight

BlockXmit Packets

```
BLOCK_CHUNK
    uint64 block_id
    uint32 block_size
    uint16 chunk_id
    uint16 chunk_size
    uint16 ack_to
    uint64 timestamp
    uint8   data[]

BLOCK_ACK
    uint64 block_id
    uint16 num_chunks
    uint64 timestamp
    BLOCK_EXTENT extents[]

BLOCK_EXTENT
    uint16 start
    uint16 count
```

Selecting Chunks

- What chunk to send next?
  - keep an estimate of the link round trip time
  - send first chunk that has not been sent within RTT
- Optional extras
  - ordered delivery can be set enabled if needed
  - each block has a priority, allowing urgent data to jump the queue

More information

- Source code, schematics etc
  - SiK firmware: http://github.com/tridge/SiK
  - 3DR Radios: http://code.google.com/p/ardupilot-mega/wiki/3DRadio
  - RFD900: http://rfdesign.com.au/index.php/rfd900
  - Block Xmit: https://github.com/tridge/cuav
  - MAVLink: https://github.com/mavlink
  - CanberraUAV: http://www.canberrauav.com/