

Presentation Outline for FX.25 Proposal

Author: James McGuire

Introduction

Jim – KB3MPL (amateur radio callsign)

Stensat Group

First satellite in orbit 2001 – deployed from Stanford's OPAL

Worked with Stanford and Cal Poly on Cubesat program inception

Cubesat components

Cansat program

Designing and building radios since a kid

Mostly working with unidirectional packet radio links

FX.25 Synopsis

FX.25 addresses an issue – AX.25 packets are easily damaged and discarded

AX.25 liabilities

Single bit error causes discard

Error recovery requires connection-based operation

Error recovery handled at top of Layer 2 or Layer 3+

No error recovery on connectionless operation

Poor performance for beacons, Tx only telemetry

FX.25 fixes this

FX.25 is an encapsulation mechanism – does not alter AX.25 packet

Add FEC capability to AX.25 basis

Error correction occurs at bottom of Layer 2

Compatible with existing AX.25 deployed user base

If you want to abandon legacy compatibility, come see me about BX.25

Provides upgrade path for future

FX.25 Details

AX.25 refresher – quickly describe AX.25 packet structure

Treat AX.25 packet as an atomic data structure – don't alter it

Important for maintaining legacy compatibility

Layer 2 function

Encapsulation and error correction at the bottom of Layer 2

FX.25 functions regardless of Layer 1 PHY considerations ... mostly

No implementation of ones-density, transition density, clock recovery, scrambling, etc.

AX.25 bit-ordering (order of transmission) is as close to Layer 1 as we get

Legacy compatibility necessitates this level of detail

AX.25 and X.25 are bit-serial protocols

FX.25 frame encapsulation components

Preamble

Correlation Tag

Why? Error tolerant outside of FEC Frame

AX.25 packet

Spec compliant if FEC Frame removed

Pad

Bit-to-Byte alignment mechanism

FEC Check Symbols

Multiple algorithms supported

Postamble

Multi-frame concatenation

Packets may be sent individually – perfectly valid

Concatenation creates large "train" of packets in single transmission

Correlation Tag function

Delineates frames

Identifies which FEC algorithm used
Multi-frame should use one Correlation Tag type per block

FX.25 Performance

AX.25 Packet basis

Minimum Null Packet size – 0x7E + 16 Header + FCS + 0x7E = 20 bytes

Bit-stuffing expansion

8 bit-shifts of 11111 that will stuff a 0 into current byte

$(256*8) / (256*256) = 1/32$ probability of inserting a stuff bit

Stuff bit expands 6/5

Nominal expansion rate is $(6/5) * (1/32) = 3.75\%$

Worst case is $6/5 = 20\%$

Note about Bit-Stuffing

Used for Packet delineation – 0x7E only valid as Flag

Not defined as a ones-density or transition-density maintenance method

Though is convenient (designers of X.25 probably thinking about it)

Transport 128 byte data field

Add minimum AX.25 overhead – 18 bytes

Expand with bit stuffing

Result – $(128 + 20) * 1.0375 = 154$ bytes in AX.25 packet

Use RS(255, 239) as example for FX.25

Pad to 239 bytes – add $(239 - 154) = 85$ bytes

FEC Codeblock is 255 bytes long

Correlation Tag adds 8 bytes

Total 263 bytes – less than 2x original

Error correction capability

RS(255, 239) has 16 Check Symbols

Can correct 8 Symbol errors

1 bit error in each of 8 individual bytes (symbols): $8/(263 * 8) = 3.8e-03$

64 bit errors in 8 bytes (symbols): $64/(263*8) = 3.04e-2$

Comparison to raw AX.25 packet

154 bytes = 1232 bits

In a $1.0e-3$ uniform error rate environment

Zero AX.25 Packets get through

All FX.25 Frames get through error-free

In a $1.0e-4$ uniform error environment

1-in-8 AX.25 Packets are errored

12.5% effective channel bandwidth reduction for unidirectional

>25% reduction if retransmission is included (retrans may be errored)

Packet collisions (e.g. APRS)

Typical operator solutions

Increase repetition rate

Increase Tx power

Both trend toward reducing the effective channel capacity

One-bit overlap

Both AX.25 Packets discarded

Both FX.25 Frames passed error free

FX.25 increases effective available bandwidth – tolerant of collisions

Efficiency – 255 bytes is a **big** packet sometimes

FX.25 supports multiple FEC methods

Reed Solomon is only the initial suggestion

Truncated Frames for RS codes

Initial support for AX.25 Packets as small as 32 bytes

FX.25 using RS(255, 239) as RS(48, 32) contains only $48+8 = 56$ bytes

Effective “Gain”

- FEC often measured in “Coding Gain”

- Error performance matches equivalent gain for un-FECd data

- Can be expressed as “Effective throughput gain” if that’s easier to understand

- No physical gain

 - Won’t magically make radio signals propagate farther

 - Will make reception possible where errors are present

Conclusion

- FX.25 provides –

 - Enhanced performance

 - Legacy compatibility

 - Interoperates with existing AX.25 equipment in field

 - Does not replicate AX.25 functions

 - Complimentary technology

 - Operation regardless of Layer 1 implementation

 - Minor issues with AX25 Layer 1 requirements

 - Expandable structure (which may or may not be a good thing)

Other FEC methods

- Alt RS codes

- Turbo codes

- Convolutional codes

- Value of interleaving