キューブサットにおける CCSDS プロトコルの採用理由とデコードソフトウェアに関する包括的調査報告

小池 貞利 (JF9SOM)

2026 年 5 月 3 日 (第 1 版)

概要

宇宙開発の民主化に伴い、キューブサット（CubeSat）をはじめとするナノサテライトの役割は、教育的な実証実験の域を超え、商用地球観測、高度な科学探査、さらには深宇宙探査へと急速に拡大しているが、これまでキューブサットの標準的な通信プロトコルとして君臨してきたアマチュア無線のパケット通信規格である AX.25 に代わり、NASA や ESA、JAXA といった世界の主要宇宙機関が参画する宇宙データシステム諮問委員会（CCSDS）が策定した標準プロトコルが、本来大型衛星向けであったものの、その優れた信頼性と効率性からキューブサットでの採用が急速に進んでいる。本報告書では、その実態等について、調査した結果をまとめたものである。

1　ナノサテライト通信における技術的転換点

宇宙開発の民主化に伴い、キューブサット（CubeSat）をはじめとするナノサテライトの役割は、教育的な実証実験の域を超え、商用地球観測、高度な科学探査、さらには深宇宙探査へと急速に拡大している[1)]。ミッションの高度化は、地上との通信において、より大容量かつ高信頼なデータリンクを求める結果となった。これまでキューブサットの標準的な通信プロトコルとして君臨してきたのは、アマチュア無線のパケット通信規格である AX.25 であったが、現代のミッション要求に対してその技術的限界が顕在化している[2)3)]。

一方で、NASA や ESA、JAXA といった世界の主要宇宙機関が参画する宇宙データシステム諮問委員会（CCSDS）が策定した標準プロトコルは、本来大型衛星向けであったものの、その優れた信頼性と効率性からキューブサットでの採用が急速に進んでいる[4)5)]。この移行は単なるプロトコルの変更にとどまらず、衛星の設計思想、地上局の運用モデル、およびセキュリティの確保という多角的な側面におけるパラダイムシフトを意味している[6)]。本報告書では、AX.25 と比較した際の CCSDS の技術的メリットを詳述し、さらに現代の衛星運用を支えるデコードソフトウェアの現状について包括的な調査結果を提示する。

1） Daniel Estévez, “gr-satellites GitHub project,” `https://github.com/daniestevez/gr-satellites` (2026 年 5 月 3 日にアクセス)

2） OpenC3, “OpenC3 COSMOS,” `https://openc3.com/` (2026 年 5 月 3 日にアクセス)

3） Marcus Müller, “AX.25 is not a good protocol,” `https://ham.stackexchange.com/questions/20770/ax-25-protocol-packet-size` (2026 年 5 月 3 日にアクセス)

4） G. Gracia-Sola et al., “S-Band CCSDS Compatible Transmitter Communications Layer for CubeSats,” `https://upcommons.upc.edu/bitstreams/3b71f473-5cc9-45a8-bde5-790b5b6666e6/download` (2026 年 5 月 3 日にアクセス)

5） Gazagnaire, “CCSDS protocol stack for CubeSats,” `https://gazagnaire.org/blog/2026-04-15-ccsds-protocol-stack.html` (2026 年 5 月 3 日にアクセス)

6） A. Reindl et al., “Security-per-Watt (SpW) heuristic for CubeSats,” `https://arxiv.org/html/2604.00303v1` (2026 年 5 月 3 日にアクセス)

2　AX.25 プロトコルの遺産と技術的負債

2.1　起源と普及の背景

AX.25 プロトコルは、1980 年代にアマチュア無線のパケット無線通信のために開発された。有線通信規格である X.25 をベースに、無線チャネルの特性に合わせて調整されたこのプロトコルは、実装が容易で、既存のアマチュア無線用端末（TNC: Terminal Node Controller）との親和性が高かった[7)8)]。初期のキューブサット開発において AX.25 が選択された理由は、性能的な優位性ではなく、入手可能なハードウェアやツールの豊富さという「利便性」に依るところが大きかった[9)]。

2.2　AX.25 の構造的欠陥と通信効率の低下

AX.25 は、本質的に「信頼性の高いチャネル」または「再送が容易な地上環境」を想定して設計されている。宇宙空間という極めてノイズが多く、ドップラーシフトやフェージングが頻発する環境では、その設計思想がボトルネックとなる。

AX.25 のフレーム構成は、HDLC（High-Level Data Link Control）に基づいており、フレームの境界を示すために「フラグバイト（0x7E）」を使用し、データ中に同じパターンが現れないよう「ビットスタッフィング」を行う。このプロセスは、データの内容によってフレーム長が変動し、受信側の処理負荷を高める要因となる[10)]。さらに、AX.25 の誤り制御は、16 ビットの CRC（巡回冗長検査）による誤り検出のみに限定されている[11)12)]。

項目	AX.25 の仕様・特徴	宇宙ミッションにおける課題
誤り制御	CRC-16 による誤り検出のみ	1 ビットの誤りでもパケット全体が破棄される
信頼性確保	Go-Back-N ARQ（再送制御）	長距離・高遅延リンクではスループットが激減する
アドレッシング	6 文字のコールサイン + 4 ビットの SSID	複雑なネットワーク構成やサービス分離が困難
物理層の独立性	未定義（主に 1200/9600bps の FSK が一般的）	高速な変調方式（QPSK, DVB-S2 等）への対応が標準化されていない
セキュリティ	標準仕様には存在しない	コマンドの偽装や盗聴に対して極めて脆弱

7）Wikipedia, “AX.25,” https://en.wikipedia.org/wiki/AX.25 (2026 年 5 月 3 日にアクセス)

8）NotBlackMagic, “AX.25 Frame Encoder/Decoder,” https://notblackmagic.com/bitsnpieces/ax.25/ (2026 年 5 月 3 日にアクセス)

9）Satellite Ground Station, “AX.25 vs CSP: When Each is Used,” https://satellitegroundstation.com/resources/ax25-vs-csp-when-each-is-used/ (2026 年 5 月 3 日にアクセス)

10）Daniel Estévez, “gr-satellites latest developments,” FOSDEM 2020, https://archive.fosdem.org/2020/schedule/event/gr_satellites_latests_developments/attachments/slides/3599/export/events/attachments/gr_satellites_latests_developments/slides/3599/slides.pdf (2026 年 5 月 3 日にアクセス)

11）QB50 Project, “AX.25 Telemetry Frame Format Guide,” https://www.scribd.com/document/328513000/Ax-25proctocol (2026 年 5 月 3 日にアクセス)

12）Marcus Müller, “Coding performance on the AX.25 radio packet,” Virginia Tech, https://vtechworks.lib.vt.edu/items/456baed2-b219-45e7-81c3-1277272b7b56 (2026 年 5 月 3 日にアクセス)

CRC による誤り検出しか持たない AX.25 では、エラーが発生した際に「再送」を要求する以外の選択肢がない。しかし、衛星通信における通信ウィンドウ（パス）は数分から十数分と限られており、再送による遅延はミッションの成否に関わる重大なリスクとなる[13)]。特に、S バンドや X バンドを用いた高速通信においては、数ミリ秒の通信断絶が数メガバイトのデータ損失に直結するため、再送に依存したモデルは限界を迎えている[14)]。

3 CCSDS プロトコルによる先進的通信アーキテクチャ

3.1 階層化された標準化の恩恵

CCSDS（Consultative Committee for Space Data Systems）プロトコルスタックは、現代のインターネットを支える TCP/IP モデルと同様に、各層が独立して定義されている。この階層構造により、ミッションの要求に応じて特定の層のみをアップグレードすることが可能となる[15)]。

CCSDS の主要な階層は、以下の通りである： 1. **物理層（Physical Layer）**：変調方式や符号化方式を定義する。 2. **データリンク層（Data Link Layer）**：TM（テレメトリ）転送フレームや TC（テレコマンド）転送フレームを管理する[16)]。 3. **ネットワーク層（Network Layer）**：宇宙パケットプロトコル（Space Packet Protocol）によるアドレッシングを行う。 4. **セキュリティ層（Security Layer）**：SDLS（Space Data Link Security）による暗号化・認証を提供する。

3.2 宇宙パケットプロトコルの柔軟性

CCSDS のアプリケーション層ユニットである「宇宙パケット（Space Packet）」は、11 ビットの APID（Application Process Identifier）をヘッダーに持つ。この APID により、衛星内の異なるサブシステムやアプリケーションごとに個別の通信ストリームを確立でき、AX.25 の SSID に比べて遥かに柔軟なマルチプレクシングが可能となる。

宇宙パケットの構造は、6 バイトのプライマリヘッダーと、オプションのセカンダリヘッダー、および最大 65,536 バイトのデータフィールドで構成される。プライマリヘッダーにはバージョン番号、APID、シーケンスカウント、パケット長が含まれ、チェックサムやリトライロジックは下位層に委ねられるシンプルな設計となっている。この「シンプルさ」は、リソースの限られた衛星搭載コンピュータでも効率的に処理できることを意図している。

3.3 転送フレームと仮想チャネルの概念

データリンク層では、宇宙パケットを格納するカプセルとして「転送フレーム（Transfer Frame）」が用いられる[17)]。ここで特筆すべきは「仮想チャネル（Virtual Channel, VC）」の概念である。一つの物理的な無線リンクの中に、最大 8 個（TM フレームの場合）の独立した論理的チャネルを構築できる。

例えば、VC0 をクリティカルな衛星状態（Housekeeping）テレメトリ、VC1 を科学データ（Science

13） NanoSats Database, “Mission Control Systems (MCS) for Nanosatellites,” https://www.nanosats.eu/tables.html (2026 年 5 月 3 日にアクセス)

14） SatNOGS Wiki, “Decode Telemetry and Packets,”(https://wiki.satnogs.org/Decode_Telemetry_and_Packets) (2026 年 5 月 3 日にアクセス)

15） Gazagnaire, “CCSDS protocol stack for CubeSats,” https://gazagnaire.org/blog/2026-04-15-ccsds-protocol-stack.html (2026 年 5 月 3 日にアクセス)

16） Gazagnaire, “Transfer Frames (Data Link Layer),” https://gazagnaire.org/blog/2026-04-15-ccsds-protocol-stack.html (2026 年 5 月 3 日にアクセス)

17） Gazagnaire, “Transfer Frames,” https://gazagnaire.org/blog/2026-04-15-ccsds-protocol-stack.html (2026 年 5 月 3 日にアクセス)

Data)、VC7 をアイドルデータといった具合に使い分けることで、緊急性の高いデータを低優先度のデータストリームから分離して送信することが可能になる。これは、単一のストリームしか持たない AX.25 と比較して、ミッション運用の確実性を飛躍的に高める機能である。

4　前方誤り訂正（FEC）とリンクバジェットの最適化

4.1　強力な符号化による通信路の安定化

CCSDS プロトコルの採用により得られる最も顕著な技術的メリットは、高性能な前方誤り訂正（FEC）技術の統合である[18)19)20)]。AX.25 がエラーを検出してパケットを捨てるのに対し、CCSDS は受信側で誤りを訂正することを前提としている[21)22)]。

CCSDS で標準化されている主な FEC 方式には、以下のものがある：* **リード・ソロモン符号（RS 符号）**：バースト誤りに強く、(255, 223) の構成が一般的である[23)]。 * **畳み込み符号（Convolutional Code）**：ビタビ復号と組み合わせて使用され、$k = 7, r = 1/2$ の構成が標準的である[24)]。 * **連結符号（Concatenated Code）**：RS 符号を外符号、畳み込み符号を内符号として組み合わせることで、極めて高い訂正能力を実現する。* **LDPC（低密度パリティ検査符号）**：シャノン限界に迫る性能を持ち、次世代の高速・深宇宙通信の標準となっている[25)26)]。

4.2　符号化利得の定量的メリット

符号化利得（Coding Gain）は、同一のビット誤り率（BER）を達成するために必要な E_b/N_0（1 ビットあたりのエネルギー対ノイズ電力密度比）をどれだけ削減できるかを示す指標である[27)]。

FEC 方式	およその符号化利得 (BER 10^{-5} 時)	特徴
未符号化 (AX.25 等)	0 dB	ノイズに極めて弱く、高出力が必要
畳み込み符号のみ ($k = 7, r = 1/2$)	5 - 6 dB	実装が容易で、広く普及している

18） bmflynn, “CCSDS — Rust library for Spacecraft Data Stream Decoding,” `https://lib.rs/crates/ccsds` (2026 年 5 月 3 日にアクセス)

19） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” `https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf` (2026 年 5 月 3 日にアクセス)

20） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” `https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf` (2026 年 5 月 3 日にアクセス)

21） Daniel Estévez, “Decoding BGM-1 GMSK telemetry,” `https://destevez.net/tag/ccsds/` (2026 年 5 月 3 日にアクセス)

22） Daniel Estévez, “Coding for HIT satellites and other CCSDS satellites,” `https://destevez.net/2017/01/coding-for-hit-satellites-and-other-ccsds-satellites/` (2026 年 5 月 3 日にアクセス)

23） altillimity, “libccsds: A library to handle several CCSDS routines,”

24） Daniel Estévez, “Coding for HIT satellites and other CCSDS satellites,” https://destevez.net/2017/01/coding-for-hit-satellites-and-other-ccsds-satellites/ (2026 年 5 月 3 日にアクセス)

25） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf (2026 年 5 月 3 日にアクセス)

26） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf (2026 年 5 月 3 日にアクセス)

27） EECS 555, “Approximate Coding Gain Comparison,” University of Michigan, http://www.eecs.umich.edu/courses/eecs555/chap9.pdf (2026 年 5 月 3 日にアクセス)

FEC 方式	およその符号化利得 (BER 10^{-5} 時)	特徴
連結符号 (RS + 畳み込み)	7 - 9 dB	長年の宇宙実績があり、非常に堅牢
LDPC 符号 / ターボ符号	9 - 11 dB 以上[28]	現代の最高性能。深宇宙通信に不可欠

この「数 dB の差」は、キューブサットの設計において決定的な意味を持つ。3dB の利得向上は、送信電力を半分にするか、あるいは通信距離を 1.4 倍に延ばすことと同義である。電力リソースが極めて限られ、かつアンテナの利得も限定的な 1U〜3U サイズの衛星にとって、CCSDS による FEC の導入はミッションの存続を左右する要素となる。

5 セキュリティの強化と「Security-per-Watt」の概念

5.1 SDLS による防護

近年のキューブサットは、安価な SDR（ソフトウェア無線）機器の普及により、第三者によるアップリンク攻撃（コマンド偽装）のリスクに晒されている。AX.25 には認証機能が存在しないため、周波数とプロトコルを知る者であれば誰でも衛星を操作できてしまう可能性がある。

CCSDS の SDLS（Space Data Link Security）プロトコルは、フレームレベルで AES-GCM などの暗号アルゴリズムを用いた認証と暗号化を提供する。これにより、以下の 3 つのセキュリティ目標を達成できる：1. **認証（Authentication）**：コマンドの送信者が正当な地上局であることを確認する。 2. **機密性（Confidentiality）**：テレメトリデータやミッションデータを傍受から保護する。 3. **アンチリプレイ（Anti-replay）**：過去の正当なコマンドを再利用して攻撃することを防ぐ。

5.2 Security-per-Watt (SpW) 経験則

キューブサットにおけるセキュリティ実装の課題は、消費電力である。研究によれば、単純な地上向け暗号化の転用ではなく、宇宙環境に最適化された軽量な暗号実装を採用することで、消費電力単位あたりのセキュリティ効果（SpW: Security-per-Watt）を最大 2.7 倍高めることが可能である。CCSDS の SDLS は、衛星の計算資源に合わせてセキュリティレベルを調整できるように設計されており、リソース制約の厳しいナノサテライトにとって現実的な選択肢となっている。

6 デコード・運用ソフトウェアの包括的調査

CCSDS の採用を後押ししているもう一つの要因は、オープンソースを中心とした強力なソフトウェアエコシステムの存在である。

6.1 デコード・ソフトウェア

衛星から受信した無線信号をデジタルデータに変換し、さらにプロトコルを解釈して人間が読める形式にするためのツールが多数存在する。

28） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf (2026 年 5 月 3 日にアクセス)

gr-satellites (GNU Radio モジュール) Daniel Estévez 氏によって開発されているgr−satellites は、アマチュア衛星および小型衛星のデコードにおけるデファクトスタンダードである。* **特徴**: GNU Radio 上で動作し、100 以上の衛星ミッションをサポートする[29)]。CCSDS の畳み込み符号、RS 符号、LDPC 符号、および TM/TC 転送フレーム、宇宙パケットのデコードを包括的にカバーしている[30)]。* **柔軟性**: SatYAML と呼ばれる YAML 形式の定義ファイルを用いて、特定の衛星の通信方式を定義できる。これにより、新規ミッションへの対応が極めて迅速に行える[31)]。

SatNOGS (Satellite Networked Open Ground Station) Libre Space Foundation が運営する世界規模の地上局ネットワークプロジェクトである[32)33)]。 * **構成**: Raspberry Pi と SDR を組み合わせた地上局クライアントが、受信データをサーバーにアップロードする[34)]。 * **デコード**: 受信したバイナリデータは Kaitai Struct を用いて定義されたデコーダーによって処理され、Web ダッシュボード（Grafana）に表示される。CCSDS ベースの衛星も多数サポートされており、クラウドソースによる運用が可能となっている。

6.2 ミッションコントロール・システム (MCS)

デコードしたデータを蓄積し、衛星の状態を監視・制御するための上位システムである。

OpenC3 COSMOS Ball Aerospace の COSMOS を起源とし、現在は OpenC3, Inc. が開発を主導している[35)36)]。 * **機能**: テレメトリの表示、グラフ化、コマンド送信、スクリプト実行（Ruby/Python）などを統合的に提供する[37)]。 * **CCSDS 対応**: cosmos−ccsds_transfer_frames などのプラグインを使用することで、CCSDS 転送フレームから宇宙パケットを抽出する処理を容易に実装できる[38)]。 cFS (core Flight System) との連携プラグインも提供されており、NASA 標準に準拠した運用環境を構築できる[39)]。

Yamcs (Yet Another Mission Control System) Space Applications Services が開発している Java ベースの MCS である[40)41)]。 * **特徴**: ISS（国際宇宙ステーション）のペイロード運用にも使用されており、CCSDS の TM/TC プロトコル、SLE (Space Link Extension)、CFDP (CCSDS File Delivery Protocol) にネイティブ対応している。

29） Daniel Estévez, “gr-satellites GitHub project,” https://github.com/daniestevez/gr-satellites (2026 年 5 月 3 日にアクセス)

30） Daniel Estévez, “gr-satellites documentation,” https://github.com/daniestevez/gr-satellites (2026 年 5 月 3 日にアクセス)

31） Daniel Estévez, “gr-satellites latest developments,” FOSDEM 2020, https://archive.fosdem.org/2020/schedule/event/gr_satellites_latests (2026 年 5 月 3 日にアクセス)

32） Corey Shields, “SatNOGS Network: Control hub for ground station clients,”(http://mstl.atl.calpoly.edu/~workshop/archive/2021/presen (2026 年 5 月 3 日にアクセス)

33） Corey Shields, “SatNOGS Network: Control hub for ground station clients,”(http://mstl.atl.calpoly.edu/~workshop/archive/2021/presen (2026 年 5 月 3 日にアクセス)

34） SatNOGS Wiki, “Official SatNOGS Software,”(http://mstl.atl.calpoly.edu/~workshop/archive/2021/presentations/Day%201/SSA:SDA (2026 年 5 月 3 日にアクセス)

35） OpenC3, “OpenC3 COSMOS,” https://openc3.com/ (2026 年 5 月 3 日にアクセス)

36） OpenC3, “COSMOS 7 Release Candidate Announcement,” https://openc3.com/news/cosmos-7-rc1 (2026 年 5 月 3 日にアクセス)

37） OpenC3, “COSMOS functionality,” https://github.com/OpenC3/cosmos (2026 年 5 月 3 日にアクセス)

38） ienorand, “cosmos-ccsds_transfer_frames gem for Ball COSMOS,” https://github.com/ienorand/cosmos-ccsds_transfer_frames (2026 年 5 月 3 日にアクセス)

39） NASA, “NASA OpenC3 COSMOS cFS Plugin,” https://github.com/nasa/cfs-cosmos-plugin (2026 年 5 月 3 日にアクセス)

40） Space Applications Services, “Yamcs,” https://www.nanosats.eu/tables.html (2026 年 5 月 3 日にアクセス)

41） Space Applications Services, “Yamcs,” https://www.nanosats.eu/tables.html (2026 年 5 月 3 日にアクセス)

6.3 開発用ライブラリ

ソフトウェアに CCSDS 機能を組み込むためのライブラリも充実している。

ライブラリ名	言語	主な機能・特徴
ccsdspy	Python	宇宙パケットデータの読み込みと解析に特化。天文学データの処理に強い[42]
spacepackets	Python/Rust	CCSDS/ECSS 規格に基づくパケットの構築と解析
libccsds	C++	転送フレームのデマルチプレクス、デスクランブル処理を提供する軽量ライブラリ[43]
ccsds-rs	Rust	高いメモリ安全性とパフォーマンスを両立。CADU のデフラグ、RS 復号に対応
libcsp	C	CubeSat Space Protocol。CCSDS ではないが、ネットワーク層として広く使用される[44][45]

7 実装上の考慮事項とハードウェアの選択

7.1 計算リソースと電力消費のトレードオフ

CCSDS の高度な符号化（特に LDPC やターボ符号）は、復号に多大な計算リソースを必要とする。かつての 8 ビットプロセッサと 128 バイトの RAM という構成では AX.25 しか動作させられなかったが、現代のキューブサットに搭載される ARM Cortex-M4/M7 クラスの MCU や FPGA であれば、CCSDS の処理は十分に可能である。

ただし、高速なデータレート（Mbps 級）を実現するためには、ソフトウェア処理では追いつかないケースがあり、専用のハードウェアアクセラレータやデジタル信号処理（DSP）コプロセッサの搭載が推奨される[46][47]。SwRI（Southwest Research Institute）などが提供する商用の CCSDS フォーマッタモジュールは、FPGA ベースで数 Mbps から数百 Mbps の処理能力を実現している。

42） Daniel da Silva, “ccsdspy: A Python package for reading CCSDS packet data,” https://heliopython.org/projects/ (2026 年 5 月 3 日にアクセス)

43） altillimity, “libccsds: A library to handle several CCSDS routines,” https://github.com/altillimity/libccsds (2026 年 5 月 3 日にアクセス)

44） libcsp GitHub, “Cubesat Space Protocol,” https://github.com/libcsp/libcsp (2026 年 5 月 3 日にアクセス)

45） libcsp GitHub, “Cubesat Space Protocol,” https://github.com/libcsp/libcsp (2026 年 5 月 3 日にアクセス)

46） IEEE Transactions on Aerospace, “LDPC for resource-constrained CubeSat,” https://ieeexplore.ieee.org/iel8/7/11031113/10891713.pdf (2026 年 5 月 3 日にアクセス)

47） SwRI, “CCSDS Solutions for Space Avionics Systems,” https://www.swri.org/markets/earth-space/space-research-technology/space-engineering/ccsds-solutions (2026 年 5 月 3 日にアクセス)

7.2 相互運用性の確保

CCSDS を採用する大きなメリットの一つは、既存の地上局インフラ（STDN, DSN, AFSCN 等）との親和性である。しかし、標準規格といえども実装にはバリエーション（オプション設定の差異）が存在する。例えば、畳み込み符号の出力順序（NASA-DSN 方式 vs CCSDS/NASA-GSFC 方式）や、スクランブラの多項式設定、RS 符号のインターリーブ深度などは、衛星側と地上側で厳密に一致させる必要がある。

8 ケーススタディ：最新ミッションにおける CCSDS の活用

8.1 月探査ミッション BGM-1 および LEV-1

SLIM 月着陸船に搭載された月面ホッパー「LEV-1」および民間月着陸ミッション「BGM-1」は、CCSDS プロトコルの現代的な応用例である。* **BGM-1**: S バンドおよび X バンドを使用し、15360 baud の GMSK 変調を採用。連結符号（RS 符号、インターリーブ深度 4）を用いてテレメトリを伝送している。* **LEV-1**: 435 MHz 帯（アマチュア帯）と S バンドの両方を使用。公式発表では CCSDS 準拠とされており、微弱な信号を確実に拾うための高性能な符号化が施されている。

8.2 深宇宙探査と CCSDS

ESA の「JUICE」ミッションや NASA の「Lucy」「DART」といった深宇宙ミッションでも、CCSDS 標準のターボ符号や LDPC 符号が活用されている。これらのミッションでは、地球との距離が極めて遠いため、符号化利得のわずかな向上がデータ回収率に直結する。キューブサットが「深宇宙用キューブサット」として発展する中、CCSDS への準拠は必須条件となっている。

9 将来展望：DTN と光通信への展開

CCSDS プロトコルの進化は止まっておらず、次世代の宇宙ネットワークに向けた標準化が進んでいる。

9.1 遅延耐性ネットワーク (DTN)

「Solar System Internet（太陽系インターネット）」構想の核となるのが、CCSDS が策定する DTN（Delay Tolerant Networking）である[48)]。これは、惑星間通信のような長遅延かつ断続的な通信環境でも、ストア・アンド・フォワード（蓄積交換）方式でデータを確実に届けるためのプロトコル群である。キューブサットが月軌道のゲートウェイや火星探査のノードとして機能する際、CCSDS 準拠の DTN 実装が必要不可欠となる。

9.2 高速光通信

電波による通信容量の限界（スペクトラムの枯渇）に対し、レーザーを用いた光通信への期待が高まっている。CCSDS は光通信向けのデータリンク規格も策定中であり、小規模なキューブサットであっても、CCSDS の枠組みの中で超高速通信を享受できる未来が近づいている。

48） Gazagnaire, “Delay Tolerant Networking (DTN),” https://gazagnaire.org/blog/2026-04-15-ccsds-protocol-stack.html (2026 年 5 月 3 日にアクセス)

10　総括

最近のキューブサットにおける CCSDS プロトコルの採用は、小型衛星が「おもちゃ」から「本格的な宇宙アセット」へと進化した証左である。AX.25 が提供したシンプルさと歴史的功績を認めつつも、現代のミッションが求める高データレート、高信頼性、強固なセキュリティを達成するためには、CCSDS というプロフェッショナルな基盤が不可欠である。

本報告で示した通り、CCSDS は単に「エラーに強い」だけでなく、仮想チャネルによる高度な運用管理、SDLS によるセキュリティの確保、および gr−satellites や OpenC3 COSMOS といった強力なオープンソースツールとの親和性という、統合的なソリューションを提供している。開発者は、ミッションの初期段階で CCSDS 準拠を選択することにより、将来の拡張性（コンステレーション化や深宇宙進出）と地上局運用の効率化という、長期的な利益を確保できる。

ナノサテライトの次の 10 年は、CCSDS という標準化された「宇宙の言語」を通じて、より多様なプレイヤーが高度な科学成果を創出する時代になるだろう。

ただ、最も最近では、CCSDS よりも更に効率よく伝送できる LoRA 変調もキューブサットに採用されて始めている。これについては、別途まとめることとする。
