

第10回NCESシンポジウム

衛星搭載ソフトウェアの構築技術 ～ FSWの加速回帰試験の実現 ～

名古屋大学 大学院情報学研究科

附属組込みシステム研究センター

研究員 高田 光隆

mtakada@nces.i.nagoya-u.ac.jp

目次

- NCESとJAXAとの共同研究の取り組み
- 背景、課題
 - 近くて遠い宇宙開発がより身近に
 - 噛み合わせ検証のフロントローディング
- 宇宙機シミュレータの種類と目的
 - TOMS（運用） vs. SVF（開発検証）
 - SIMULUS , HLA/RTI, Basiles
- 衛星加速試験のシミュレータ
 - HLAを用いた分散シミュレーション
 - 通信シミュレーションの高速化手法
- 今後の方向性

NCESとJAXAとの共同研究の取り組み

JAXAとの協力協定に基づき

スペースソフトウェアプラットフォーム連携研究開発拠点 をNCESに設置(2011)

年度	題目	主な成果物
2011～2014	宇宙機向けソフトウェアプラットフォームの研究開発	SpWリアルタイム性保証手法ガイドライン、SpaceWireOS
2015～2018	SpaceWire高速化のための次世代ソフトウェアレイヤ	SpaceWireOSターゲット拡張、SOIS, cFS/CFE検討
2019～2021	衛星搭載ネットワーク・ソフトウェア構築技術	SpW IF 標準化試作、SpW HAL 検討・試作
2022～2024	衛星搭載SWの加速回帰試験・ネットワークの自在化	SIB2 – SpW HAL連携、SSSP検討・試作
2025～	加速回帰試験シミュレータの実現	HALベースの加速シミュレータ

背景: 近くて遠い宇宙開発がより身近に

これまで(単品開発)

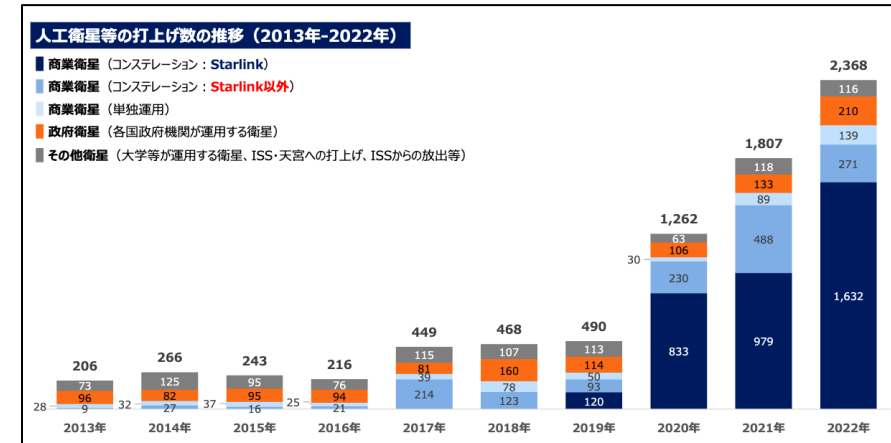
- 科学技術衛星などプロジェクトごとの開発
- 放射線対策などの宇宙グレード品による高価な部材を利用
- 採用メーカーごとに提供されるOSやミドルウェアの上でミッションアプリケーションを開発

現在、これから(量産開発)

- 国内でもJAXAだけでなく、民間によるロケット開発が進んでいる
- 打ち上げの相乗り、ISSからの放出など大学や民間によるSmallSat(超小型衛星)やコンステレーション衛星により打ち上げ数が急伸
- 車載など民生量産品を使って衛星コストを低減
- 民間の衛星ビジネス(衛星データ、衛星通信)が立ち上がり



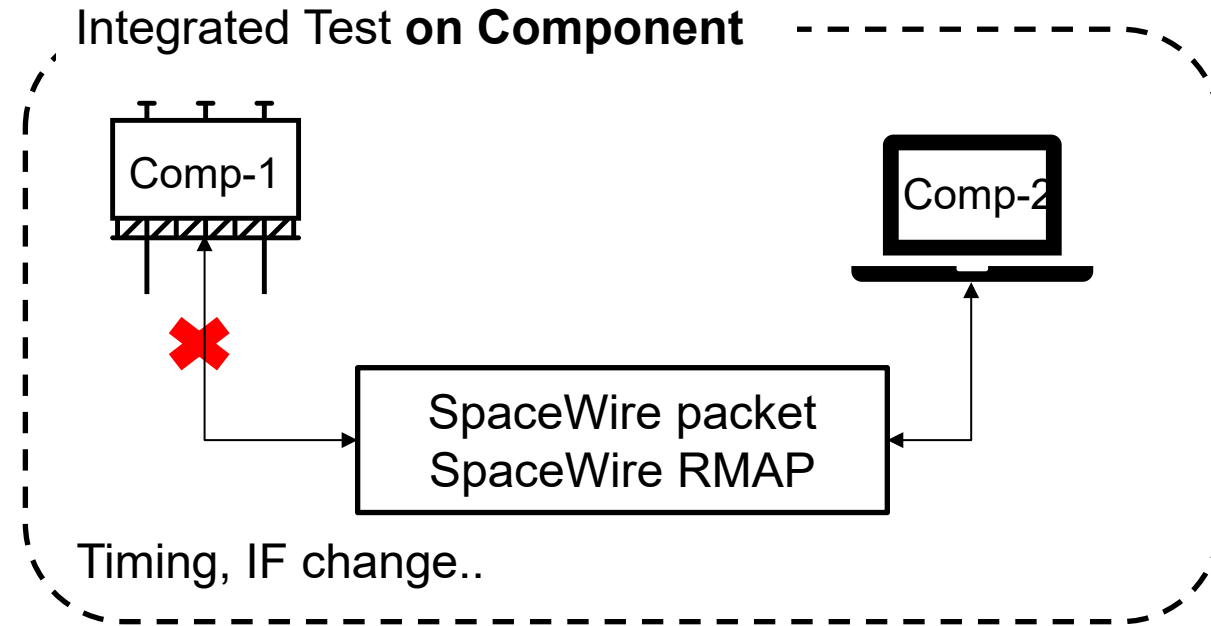
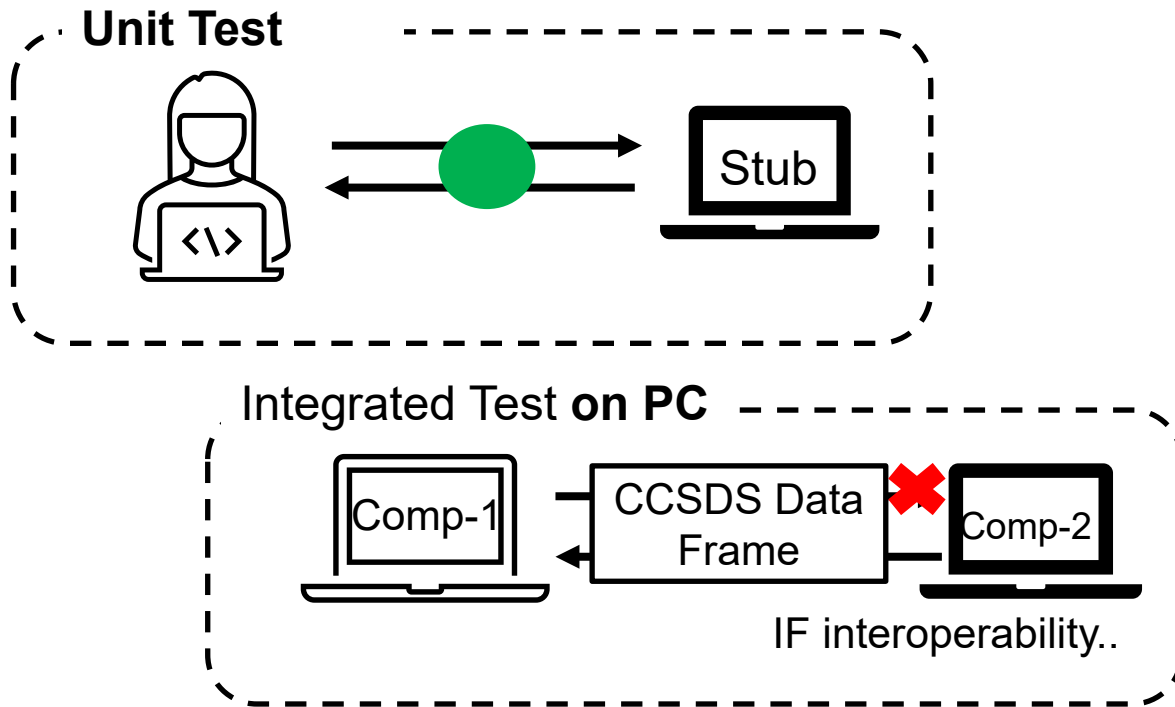
打ち上げサイクルが短く、継続的な宇宙機開発



内閣府: 宇宙輸送を取り巻く環境認識と将来像より

課題：噛み合わせ検証のフロントローディング

- 衛星開発はEM/FM の実機が揃うのが遅く、噛み合わせ検証が後工程に集中
- 手戻りしやすい開発環境として **モデル→単体→結合**を通して使えるシミュレーション環境が望まれる
- 結合テストには通信ログを回帰試験として繰り返し流す



EM : Engineering model, FM : Flight model

宇宙機シミュレータの種類と目的

主要目的	TOMS Training, Operations and Maintenance Simulator(運用)	SVF Software Validation Facility(開 発検証)
リアルタイムの動作	ソフトリアルタイム 「できるだけ速く」	「決定論的に実行できればOK」
デバッグとアーカイブ(ログ)機能	「ほとんどない」と予想(ランタイムのみ?)	広範囲で必須
サブシステムモデリングの忠実度	現実的 か 十分に典型的	OBSを検証するために「十分な忠実度」
環境のモデリング	GNC/EPS/TCSに対する主要な外部影響を備えた閉ループGNC	GNC だけがあれば良い
外部インタフェース	宇宙ー地上インタフェースと地上局をモデル化する	テレメトリとテレコマンドを流す
主なソフトウェア Framework(ESA)	SIMULUS	Basiles

- ESAではTOMS, SVFで分かれて研究が行われてきた
- 両方を満足できるようなシミュレータを検討

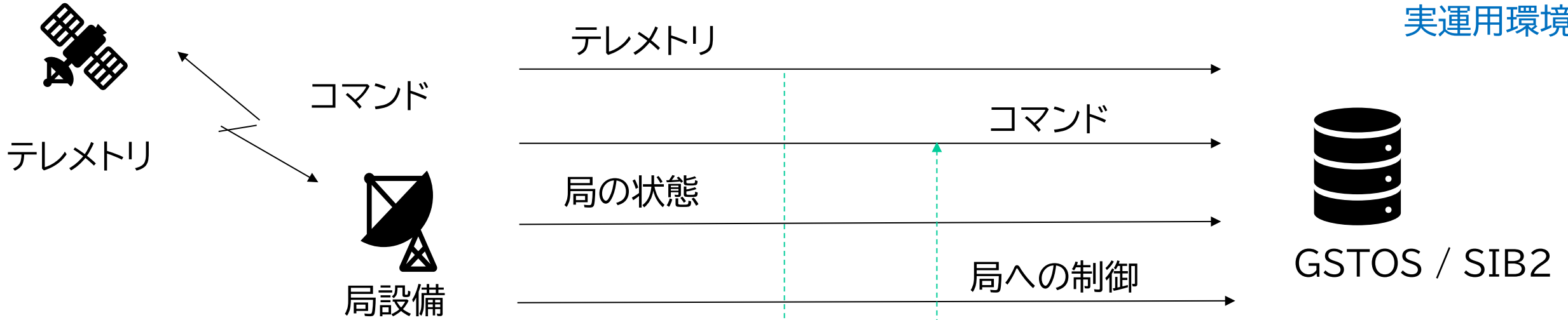
※ SESP2025のcommunity展示より筆者記

SIMULUS , HLA/RTI, Basiles

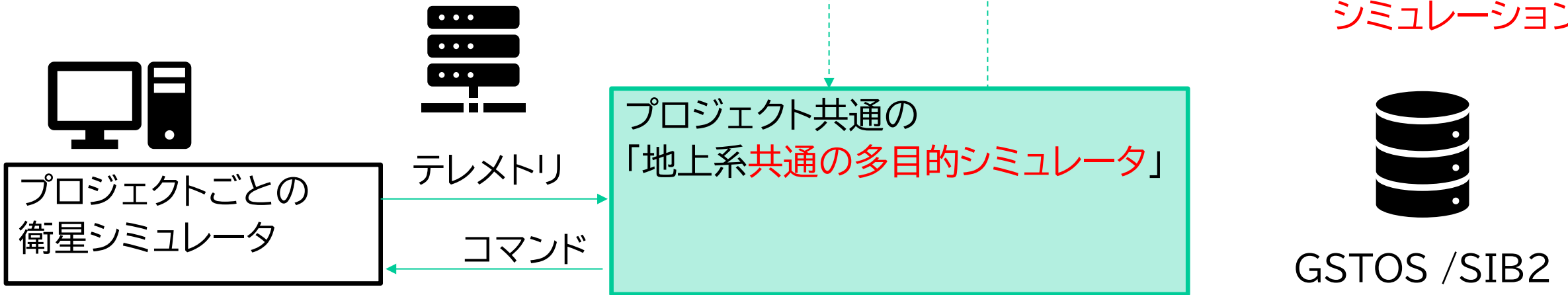
	SIMULUS	HLA/RTI	Basiles
概要	ESA(欧州宇宙機関)開発のシミュレーションフレームワーク	分散シミュレーションの国際標準(IEEE 1516)	CNES(フランス国立宇宙研究センター)開発のシミュレーション基盤
開発元	ESA	IEEE (RTI実装は複数)	CNES
主な用途	宇宙システムのシミュレーション	多分野における分散シミュレーション(主に軍事)	宇宙機やミッションシミュレーション
アーキテクチャ	統合型シミュレーションフレームワーク	RTIを介した分散シミュレーション	分散シミュレーション基盤、HLA対応
通信方式	内部コンポーネント間で同期処理	RTI(Runtime Infrastructure)を介した通信	HLAベースの通信方式を採用
標準規格	ESA独自仕様	IEEE 1516 (HLA)	HLA準拠
拡張性	ESA向けに特化、外部連携は限定的	高い(異種シミュレーション連携が可能)	宇宙プログラム間でのシミュレータ共有を想定

検討衛星シミュレータ

実運用環境



シミュレーション環境



※ GSTOS : Generic Spacecraft Test and Operations Software
※ SIB2 : Spacecraft Information Base vresion2

HLAの概念図

• フェデレート

- シミュレーションに参加する個々の構成要素（プログラム）
- フェデレート内ではフライトソフトなどのアプリケーションが動作する

• RTI

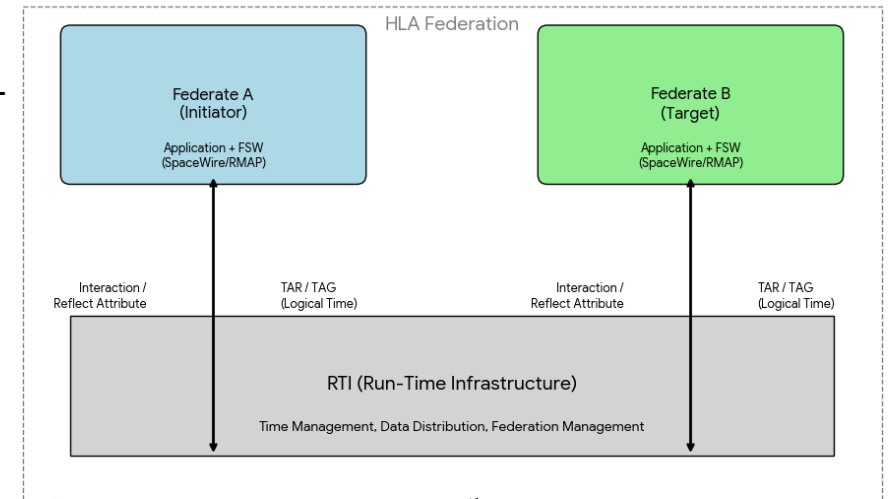
- フェデレート間のデータ交換や時刻同期を管理するシミュレーション用バス
- 個々のフェデレートはRTIを介して情報（メッセージ）をやり取りする
- メッセージを時間順にソートし、フェデレート時刻に矛盾しないように転送する

• フェデレーション

- RTIと全フェデレートを合わせてシミュレーション全体を指す

• 時刻管理

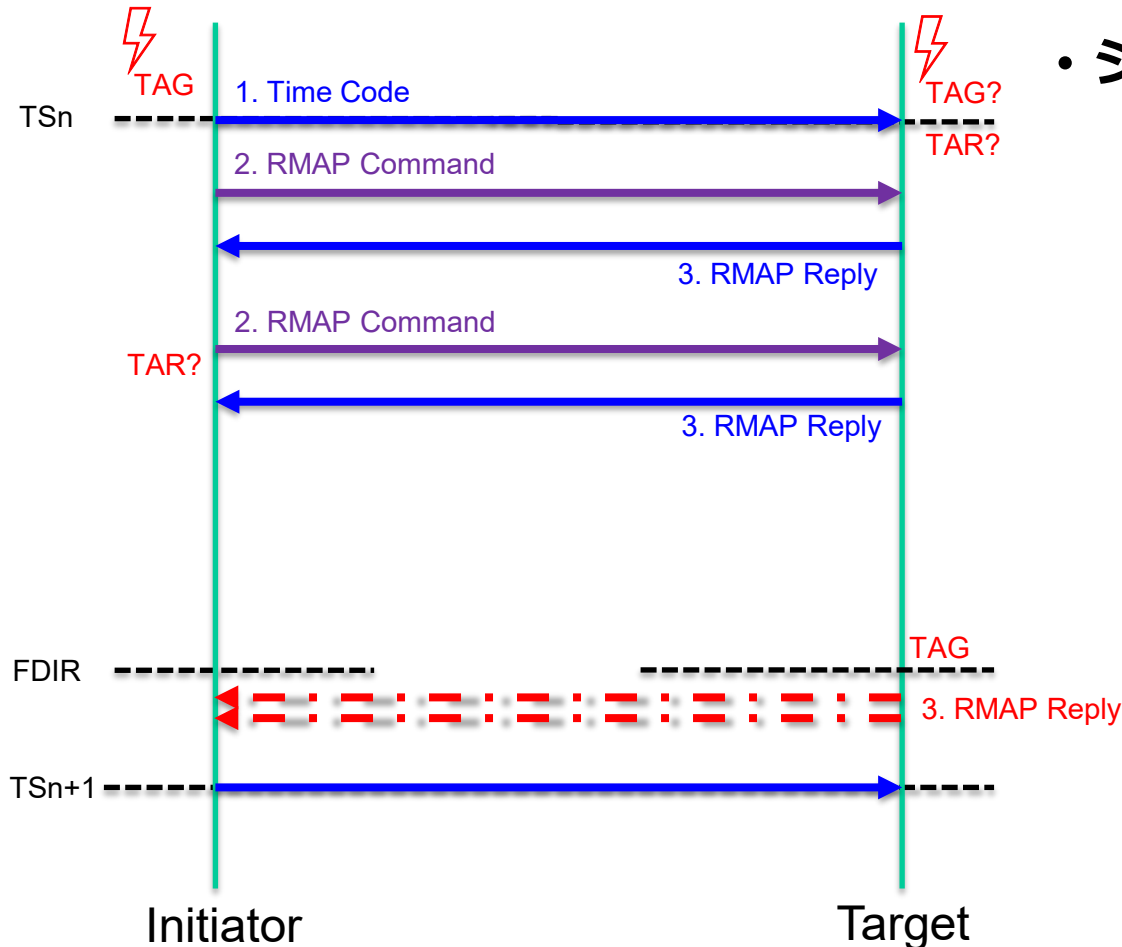
- TAR:フェデレートがRTIに対して時刻を進める要求を出す
- TAG:RTIがフェデレートに時刻を進める許可を出す



メッセージはシミュレーション時刻（HLA時刻）よりも先（未来時間）を指定する必要がある
高速化にはTAR/TAG間が広いほどよいが、通信応答（リプライ）は過去に送ることができない

HLAを通信シミュレーションに適用した場合の課題

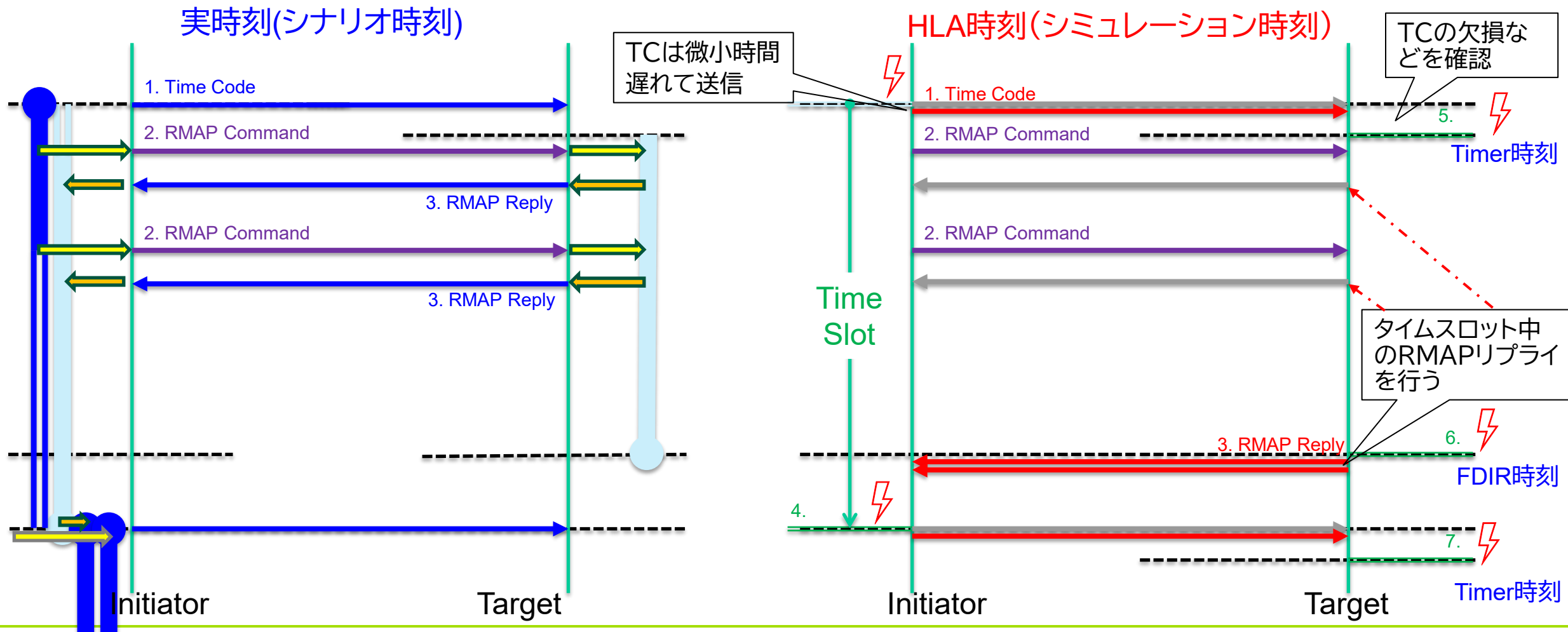
- ・受信処理を開始する時間
 - ・フェデレートが次に進む時間
- 同期点が不定



- **実機**の場合
 - 割込みハンドラが受信バッファに格納
 - スロット区間中、受信タスクが応答処理を行う
 - **シミュレータ(SILS)**の場合
 - 固定時間で時間を進める（ポーリング）
 - 何も処理しない時間が存在する
 - 時間分解能がシミュレーションの精度になる
 - フェデレートが時間を進める（HLA）
 - メッセージの到着時間が不定
 - Time Codeの遅れ、欠損が判断できない
 - FDIRまでには全メッセージは到着するが、応答処理は同スロットで返すことができない
- Reply
- ↓
- 同期点をFDIRとし、応答メッセージの時刻をHLAとは別手段で伝達するで対応

HLA通信シミュレーションの高速化手法

TAR/TAGの回数を少なくしつつも、通信シーケンスのエラーケースが実現できるタイミングを検討
Initiator : Time Codeの先頭、 Target : Time Codeタイムアウト時刻(Timer時刻)、 FDIR時刻の3点に設定
メッセージの時刻はHLA時刻とは別にSSSPでカプセル化することで過去のシナリオ時刻の送信が可能に



まとめと今後の方向性

衛星加速試験シミュレータ

- Java実装HLAライブラリを使って、実時間の4 – 5 倍の加速試験を実現
- 使用ライブラリのプロファイリングを行い、さらなる高速化へ
- 正常・異常系ケースなどのシナリオ ドキュメントの拡充
- テストシナリオの充実を図りプロジェクトへの導入

過去の成果物との連携

- SpaceWireOS を本シミュレータの基盤アプリとして拡張
- より実機適用に近いシミュレータの環境を提供

Contacts

Please feel free to ask us if you have any questions.



**Center for Embedded Computing Systems,
Nagoya University Tel : 052-789-4228 Fax:
052-789-4273**

**URL: <http://www.nces.i.nagoya-u.ac.jp/>
email: nces-office@nces.i.nagoya-u.ac.jp**

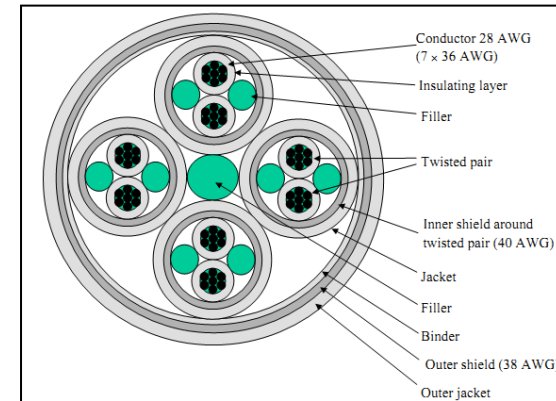
NCES SpaceWire

検索

補足資料

SpaceWireの特徴(1/2)

- 宇宙機内の機器間データ通信インターフェース規格
 - ESA（欧州宇宙機関）がIEEE1355規格をカスタマイズして提案(1994) (ECSS-E-ST-50-12C)
 - LVDSによるノイズに強い高速シリアルライン
- 幅広い転送速度（2-400Mbps）
 - データ転送レートが可変
 - Gbps向けのSpaceFibre規格も登場
- 可変長のパケットサイズ
 - データサイズの規定は無い
- 簡単なプロトコル（省リソース, 省電力）
 - FPGA実装のノードやルータも多数
 - 上位プロトコル(RMAP, SpaceWire-D, SpaceWire-R)があり、HW化の研究
- 小中型衛星では採用が進んでいる（大型のSmallSat(8U以上)でも搭載が始まっている）



SpaceWireの特徴(2/2)

- タイムコードによるノード間の時刻送信・同期機能
- メッシュ型のネットワーク
 - 冗長経路を持つことで通信の障害耐性が強い
 - ノードを組み上げていくときに柔軟なネットワークが不可欠
- ワームホールルーティング ルータを介して接続
 - ルータ内のルーティング遅延がほとんどない
 - しかし一度**パケット衝突が発生すると、ネットワークをブロック**する恐れ

大きな
問題点

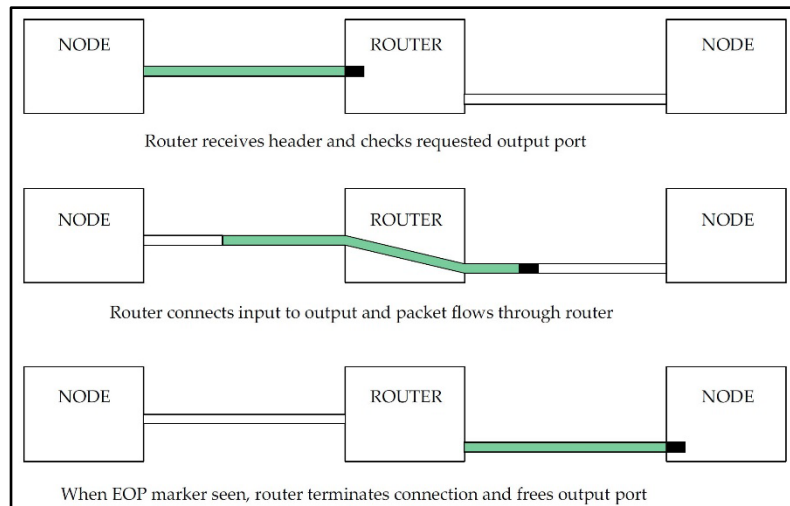


図.ワームホールルーティングの例(ECSS-E-ST-50-12Cより)

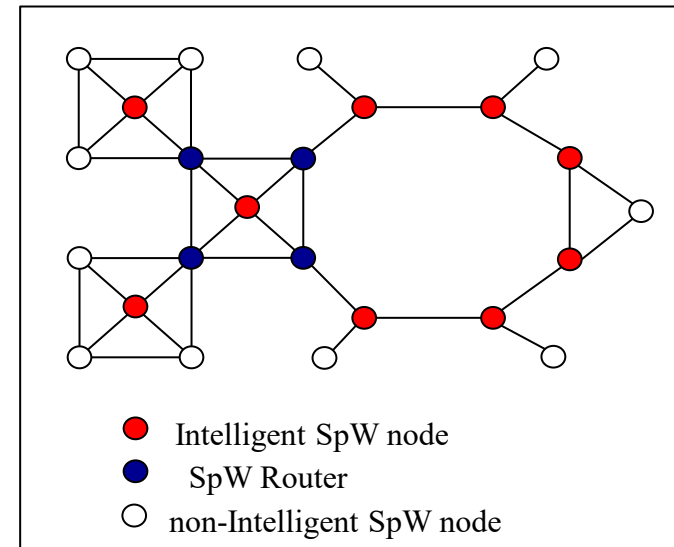


図.SpaceWireネットワーク構成例

SpaceWireの主な上位層プロトコル

RMAP : Remote memory access protocol(ECSS-E-ST-50-51C)

- SpaceWireの上位レイヤに位置する通信プロトコル
- ネットワーク越しのノードに対して、メモリマップドI/O機能でアクセスできる仕組みをそなえたプロトコル

CCSDS: Consultative Committee for Space Data Systems

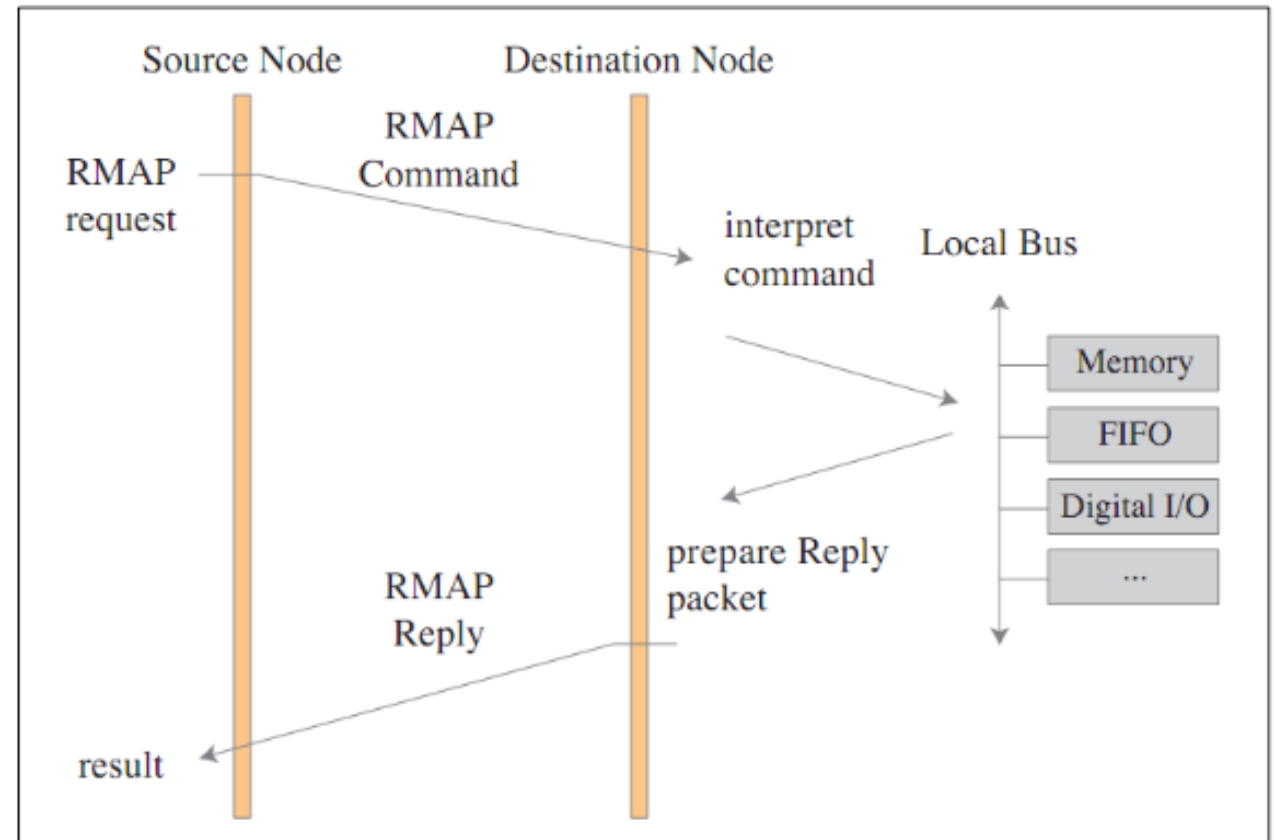
- 宇宙データシステム諮問委員会で決められた
- 主に地上系とのコマンド・テレメトリ通信で利用
- SpaceWire-PTPとも呼ばれる(ECSS-E-ST-50-51C)

SpaceWire-D : Deterministic(本研究)

- タイムコードを利用した時分割通信方式
- 時分割されたスロットに設計で予め送信する

SpaceWire-R : Reliability(JAXAが提議)

- トランスポート層(一部セッション層)の役割を担う
- データの多重化, データの分割/復元, フロー制御
- シーケンスチェック, ハートビート(オプション)



SpaceWire ミドルウェアの開発実績

SpaceWireOS(ISC2013,宇科連56,57)

- MHI製SOI –SOC向けに名大で開発
- TOPPERS/ASP,HRP2
- RMAP, SpaceWire-D
- **SpaceWire HAL実装確認のため同一デバイス(SpacePi)のTOPPERS環境へ**



@シマフジ電気

SpaceWire RMAP Library(ISC2022,宇科連66)

- POSIX API向け(C++)、地上でのPoC利用
- RAW, RMAP, SpaceWire-R
- **SpaceWire HAL実装確認のため同一デバイス(SpacePi)のLinux環境へ**
- <https://github.com/yuasatakayuki/SpaceWireRMAPLibrary>



@シマフジ電気

SBN(cFS)(FSW2017,2018)

- POSIX, RTOS(TOPPERS/HRP2を移植)
- MHI製SOI –SOC, SpacePi
- SpaceWire-RAW対応実装と通信確認を行った



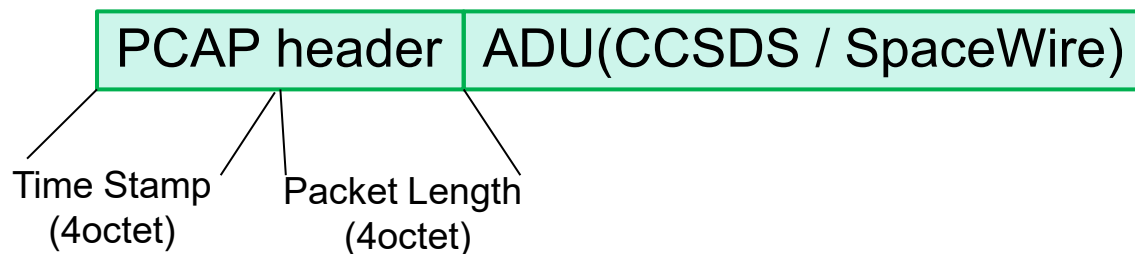
Spacecraft System Simulation Protocol(SSSP)

- 概要

- ADU(Application Data Unit)にPCAPヘッダを付与してEthernetで伝送
- ADUにはSpace Packet Protocol(CCSDS) と SpaceWireを想定
- PCAPに送信時のシミュレーション時刻を格納する

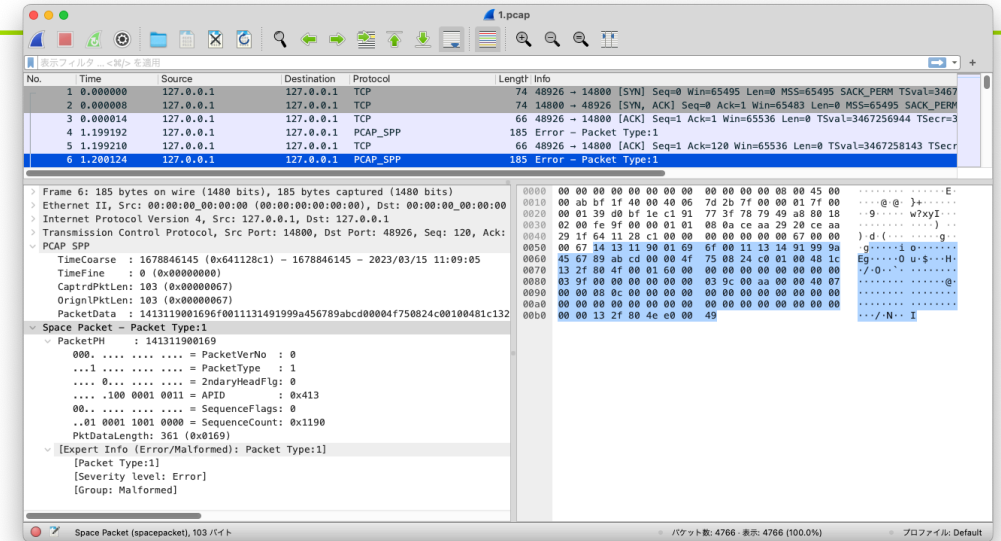
- 特徴

- SpaceWire 伝送プロトコルの可視化
- 実装が簡易のため、シミュレータ模擬環境やコンポーネントに変換器を接続することでHILSやSILSによる利用を想定
- WireShark/tcpdumpなど既存のネットワークロギングツールを利用



SSSPの開発・実証(2023-)

- 実装プロトタイプとして**pcap-nc**が公開
 - <https://github.com/matsuzaki-keiichi/pcap-nc>
 - nc コマンドをベースにSSSPを送信する
 - WireSharkの可視化プラグインも開発



- NCESでは既存研究成果（搭載ソフトウェア）をベースにSSSPを適用
 - pcap-ncとの相互疎通通信を確認
 - アプリの通信IFは既存の研究成果（RMAPLib, SIB2Gen）を利用してPC、ラズパイでのシミュレーション環境を構築

