NCES

第7回NCESシンポジウム

衛星搭載ネットワーク・ソフトウェア アーキテクチャの構築

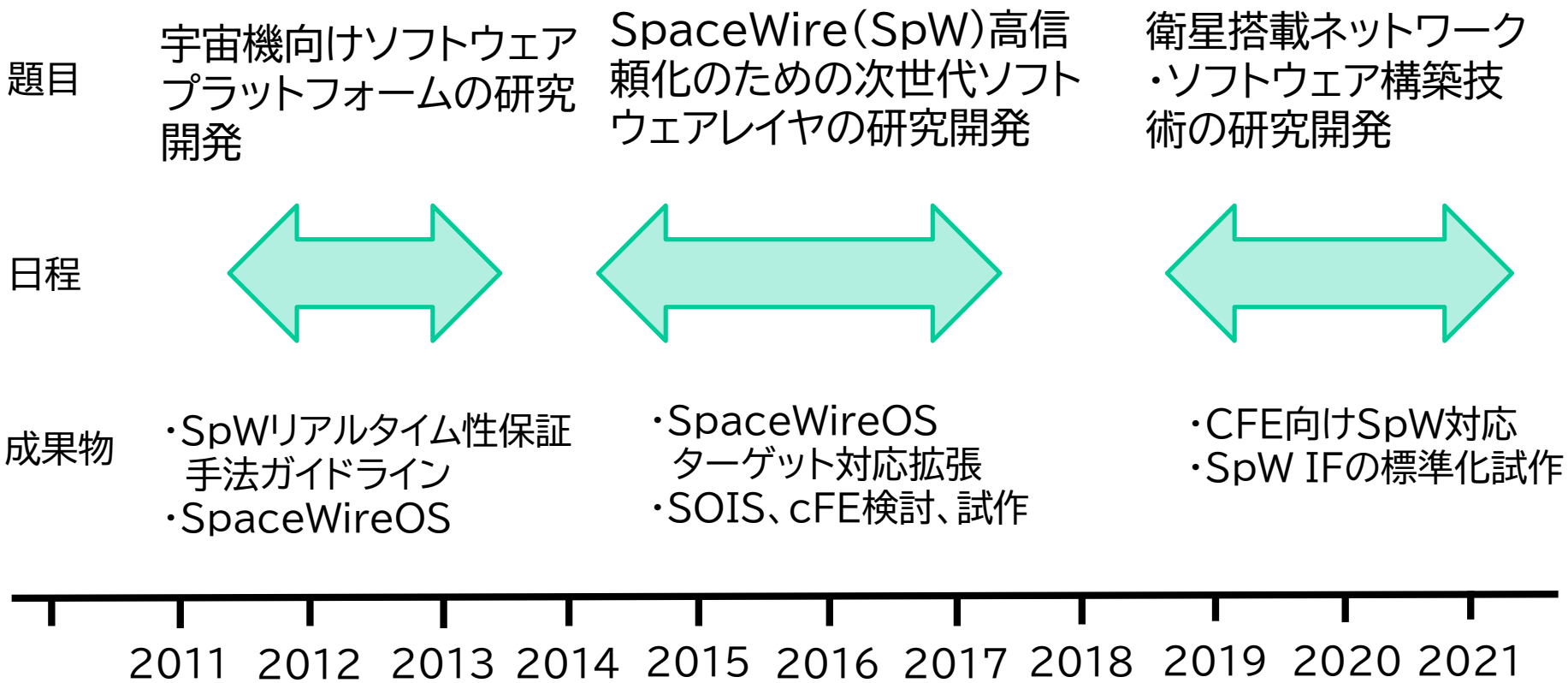
名古屋大学 大学院情報学研究科
附属組込みシステム研究センター
研究員　高田 光隆

目次

1. JAXAとの共同研究の取り組み
2. 宇宙機向けソフトウェアアーキテクチャ(SpaceWireOS)
3. ESA,NASAによる標準インタフェースとプラットフォーム実装(SOIS、cFE/CFS)に関する調査と検討
4. 民生機器(RaspberryPi)を用いたSpaceWire環境の構築
5. 車載向けソフトウェアプラットフォームとの比較
6. 今後の取り組み

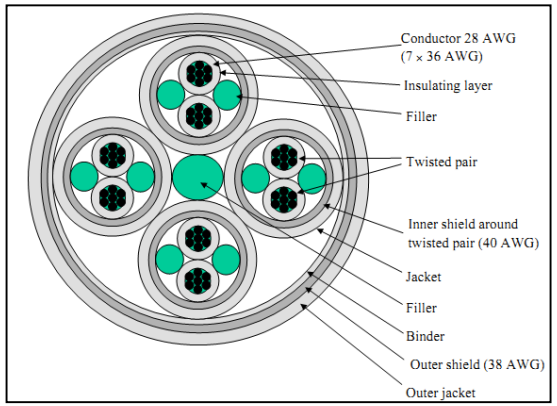
JAXAとの共同研究の取り組みと成果

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）



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
- 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
- 中大型衛星では採用が進んでいる(大型のSmallSat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同期
- 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

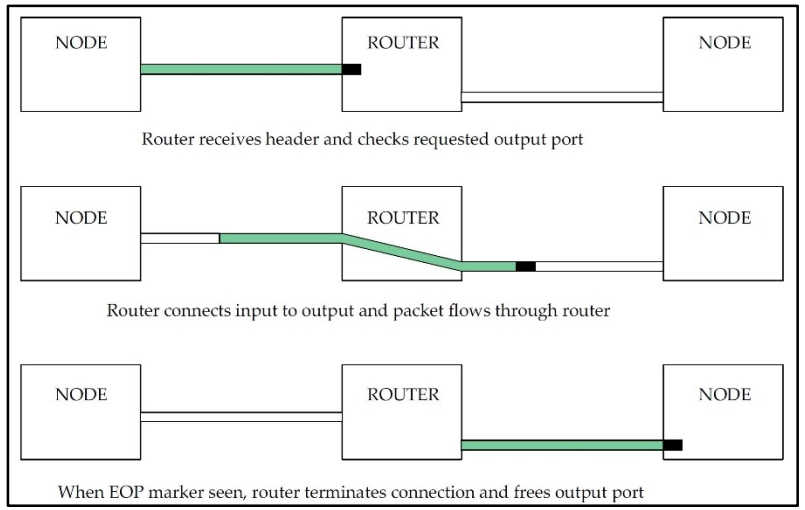


図.ワームホールルーティングの例(ECSS-E-ST-50-12Cより)

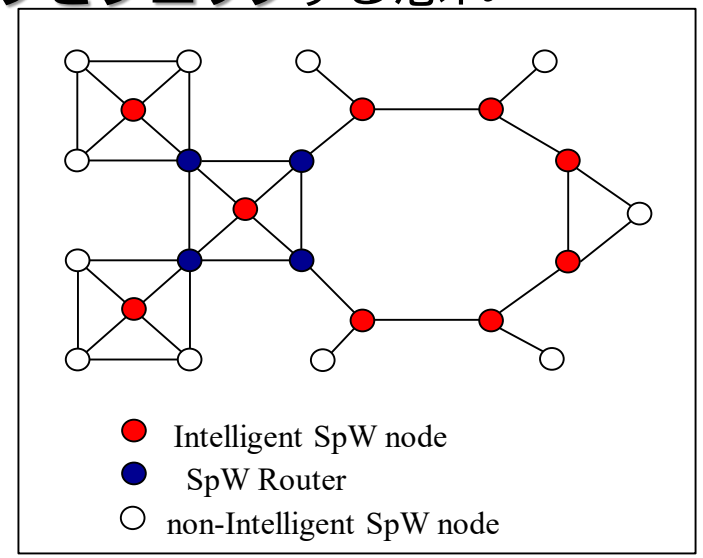
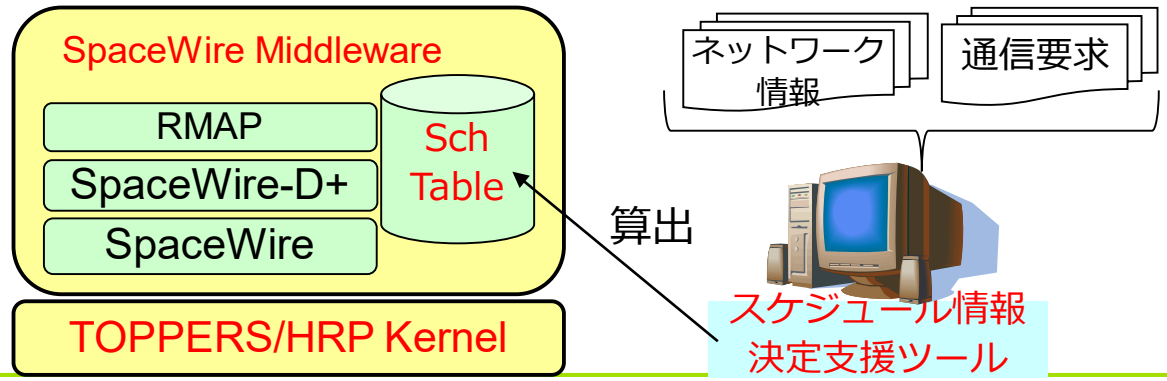


図.SpaceWireネットワーク構成例

SpaceWire OS

SpaceWireを用いた通信アプリケーションの開発において、
リアルタイム性を確保し、簡便に作成できるソフトウェアプラットフォームを構築を目指す

1. 高信頼なリアルタイムOS
   - TOPPERS/HRPカーネル,テストスイート(TTSP)を開発
2. SpaceWireミドルウェア
   - SpaceWireの問題点を解決するプロトコルスタックの開発
   - SpaceWire RAW, RMAPパケットを変更せずに使用する
3. スケジュール情報決定支援ツール
   - リアルタイム性保証を確保するスケジュール決定を支援する外部ツール



SpaceWireOSの概略図

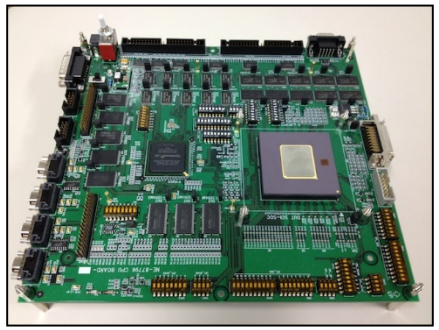
SpaceWireミドルウェア

SpaceWire-D(時分割)プロトコルの一部を拡張したソフトウェアスタック

- スロットごとにネットワーク空間を分割・設計
- 柔軟なタイムスロットを設計
- マルチスロットトランザクションの導入

JAXA、MHIで開発したSOI-SOCによる実装確認

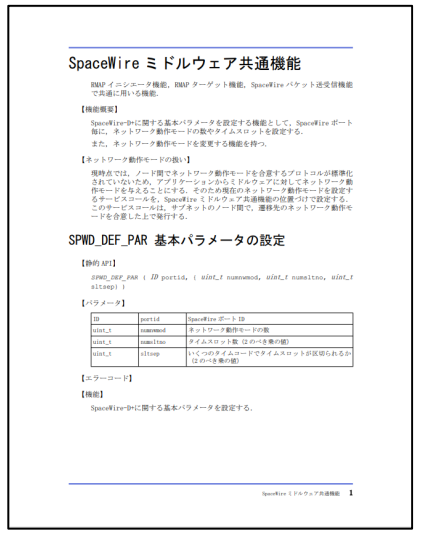
- 実装にあわせて、ミドルウェアAPI仕様とデバイスドライバIF仕様を作成
- その後のハードウェア化に寄与



(MHI製SOI-SoCボード)


SpaceWire ミドルウェア API 仕様

Center for Embedded Computing Systems, Nagoya University 0.2.0
2013 Center for Embedded Computing Systems, Nagoya University


SpaceWire ミドルウェア共通機能

RMAP イニシエータ機能, RMAP ターゲット機能, SpaceWire パケット送受信機能で共通に用いる機能.

【機能概要】

SpaceWire-D に関する基本パラメータを設定する機能として, SpaceWire ポート毎に, ネットワーク動作モードの数やタイムスロットを設定する.

また, ネットワーク動作モードを変更する機能を持つ.

【ネットワーク動作モードの扱い】

現時点では, ノード間でネットワーク動作モードを合意するプロトコルが標準化されていないため, アプリケーションからミドルウェアに対してネットワーク動作モードを与えることにする. そのため現在のネットワーク動作モードを設定するサービスコールを, SpaceWire ミドルウェア共通機能の位置づけで設定する. このサービスコールは, サブネットのノード間で, 遷移先のネットワーク動作モードを合意した上で発行する.

SPWD_DEF_PAR 基本パラメータの設定

【静的 API】

SPWD_DEF_PAR (*ID* portid, { *uint_t* numnwmod, *uint_t* numsltno, *uint_t* sltsep) }

【パラメータ】

ID	portid	SpaceWire ポート ID
uint_t	numnwmod	ネットワーク動作モードの数
uint_t	numsltno	タイムスロット数 (2 のべき乗の値)
uint_t	sltsep	いくつのタイムコードでタイムスロットが区切られるか (2 のべき乗の値)

【エラーコード】

【機能】

SpaceWire-D に関する基本パラメータを設定する.

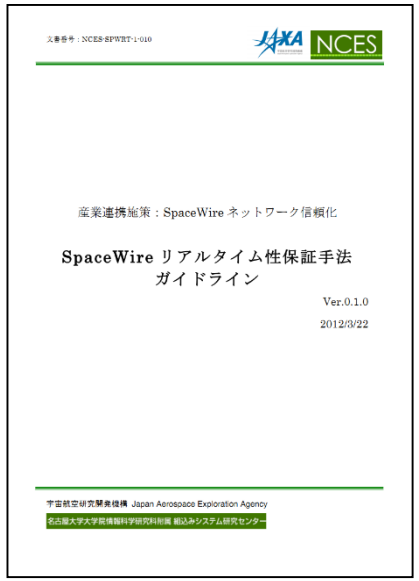
SpaceWire ミドルウェア共通機能 1

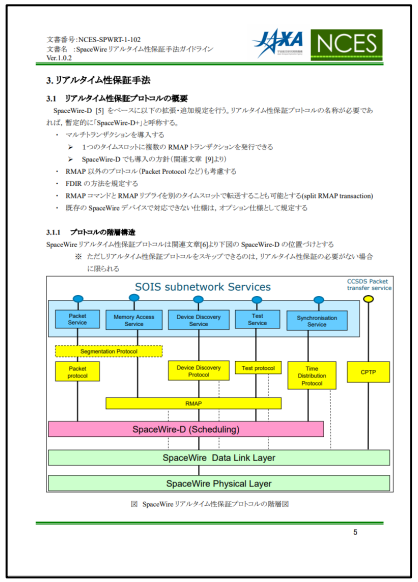

SpaceWire OS デバイスドライバ
インターフェース

Center for Embedded Computing Systems, Nagoya University 0.2.0
2013 Center for Embedded Computing Systems, Nagoya University

リアルタイム性保証手法ガイドライン

- SpaceWireを搭載する際のネットワークの設計ガイドライン(注意点)を策定
- JAXA SpaceWireオンボードサブネットワーク設計標準の適用文章(JERG-2-432)
- 一部の衛星システムのネットワーク設計に適用

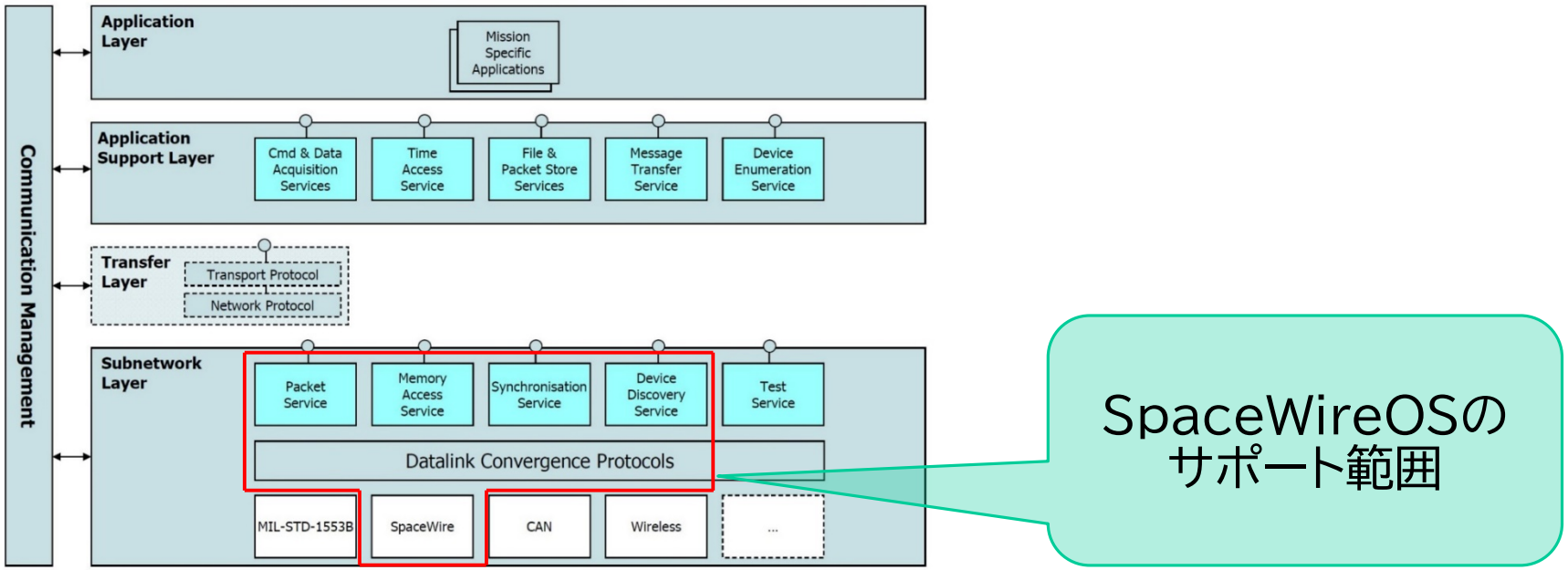




SOIS

SOIS(Spacecraft Onboard Interface Services)

- CCSDS(宇宙データシステム査問委員会)で検討しているソフトウェアのインターフェースサービス群 (CCSDS 850.0-G-2)※
- SpaceWireOSは現状Subnetwork Layerの一部のみサポート



CCSDS 850.0-G-2 より

※ https://stage.tksc.jaxa.jp/ccsds/index.html

SOISを利用するメリットと課題

- CCSDSに基づくソフトウェアインタフェースの勧告(→標準インタフェース)
- EDS(電子データシート)によってXMLデータで受け渡しが可能(CCSDS 876.0-B-1)
- 国際協調での衛星開発で利用される流れ
- JAXAのSpWオンボードサブネットワーク標準もSOISの対応をまとめている

課題

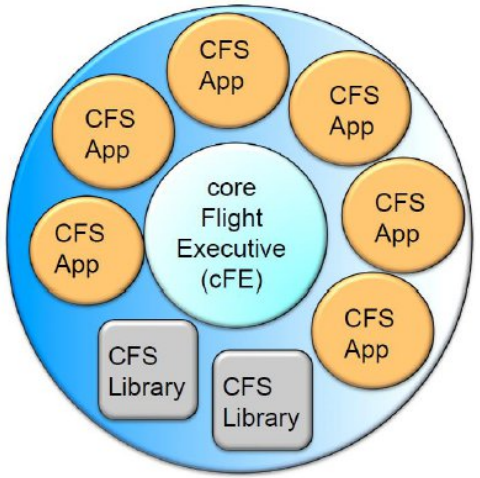
1. SOISに対応しているオープンなソフトウェアプラットフォーム例が少ない
   - ESAではSAVIOという研究プロジェクト
   - NASA/GSFCを主導する**core Flight System(CFS)**が オープンソース実装
2. 実装インタフェースがターゲットごとに異なる
   - SOISで規定しているIFの抽象度が高いため

cFE/CFSの概要

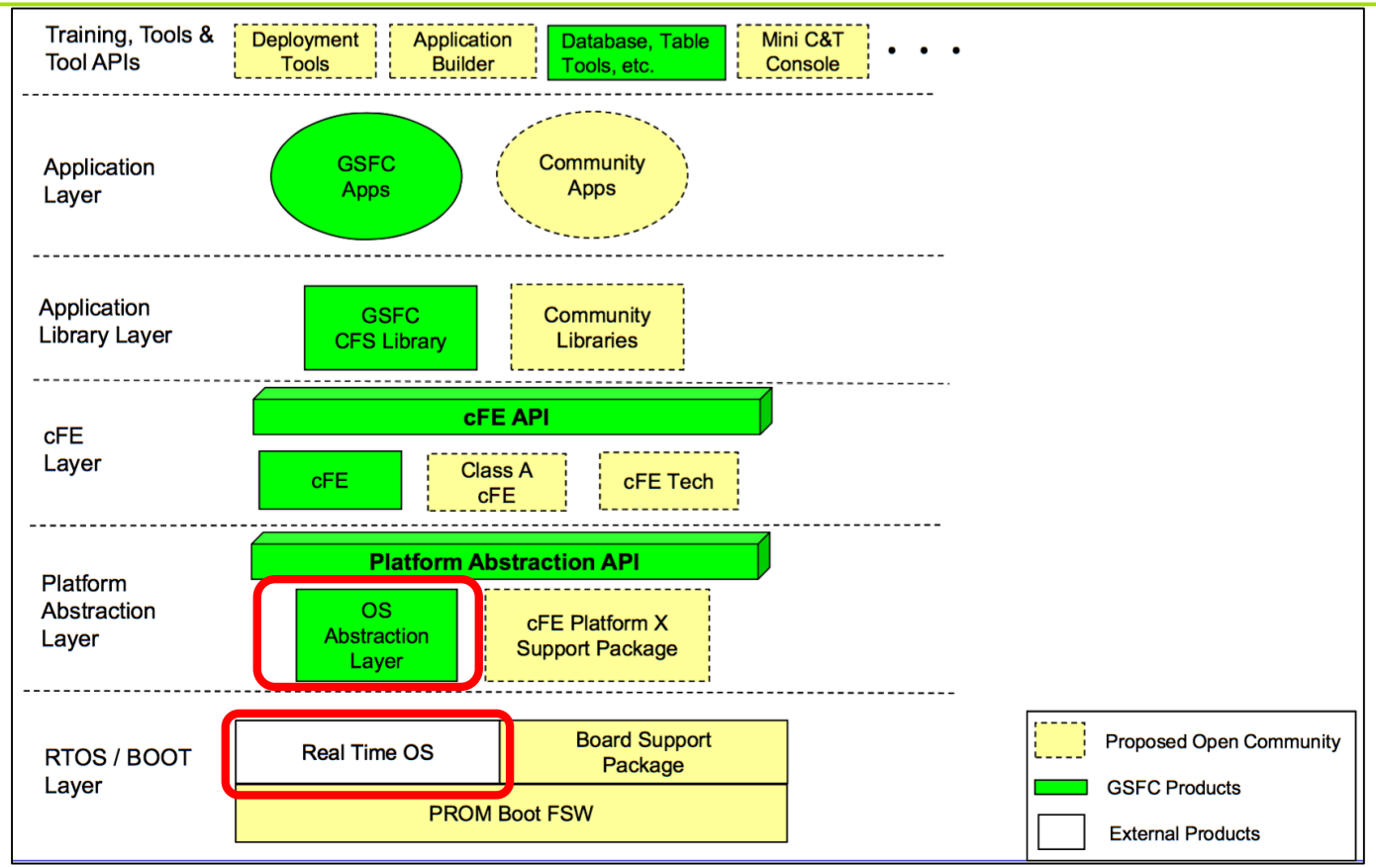
core Flight Executive / core Flight System

- NASA/GSFCが開発したOSSの宇宙機向けソフトウェア基盤
  - https://cfs.gsfc.nasa.gov
- 再利用性の高い機能モジュールをコアサービスとして提供(cFE)
  - Executive Service(ES)　：初期化, アプリ管理
  - Software Bus(SB)　：メッセージ通信
  - Event Service(EVS)　：アプリエラーメッセージ送信
  - Table Service(TS)　：パラメータ管理
  - File Service(FS)　：ファイル操作
  - Time Service(TIME)　：時刻管理
- NASA衛星でのフライト実績がある

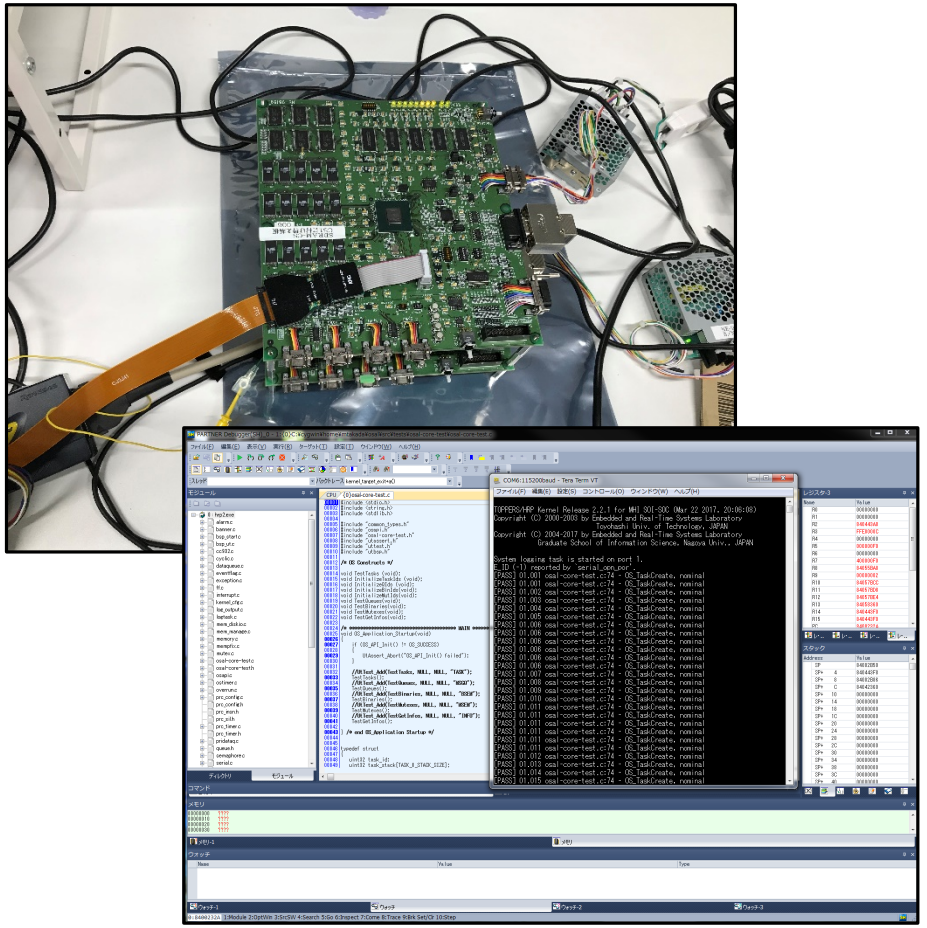
すべてがオープンソース
SpaceWireOSをCFSへ移植



cFE/CFSの調査とSpaceWireOSの移植対応



CFS Development Guideより

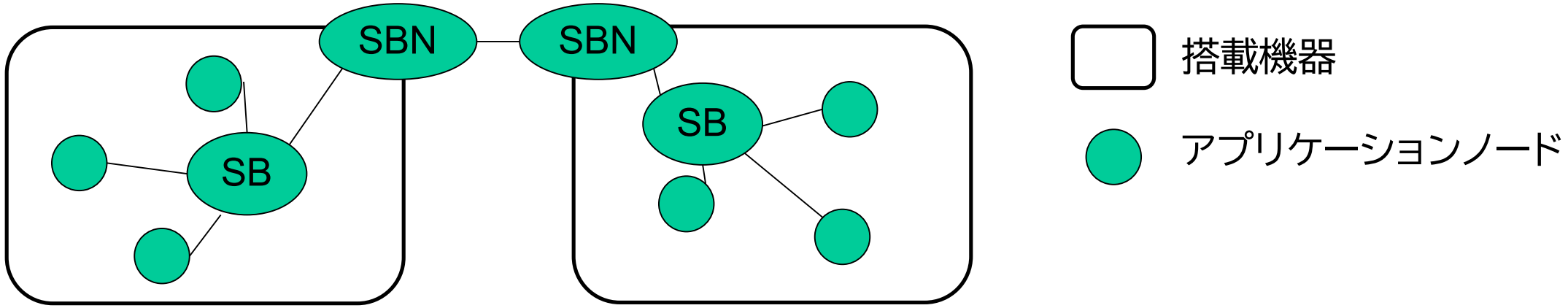


SOISOC上でcFEのアプリケーション動作確認の様子

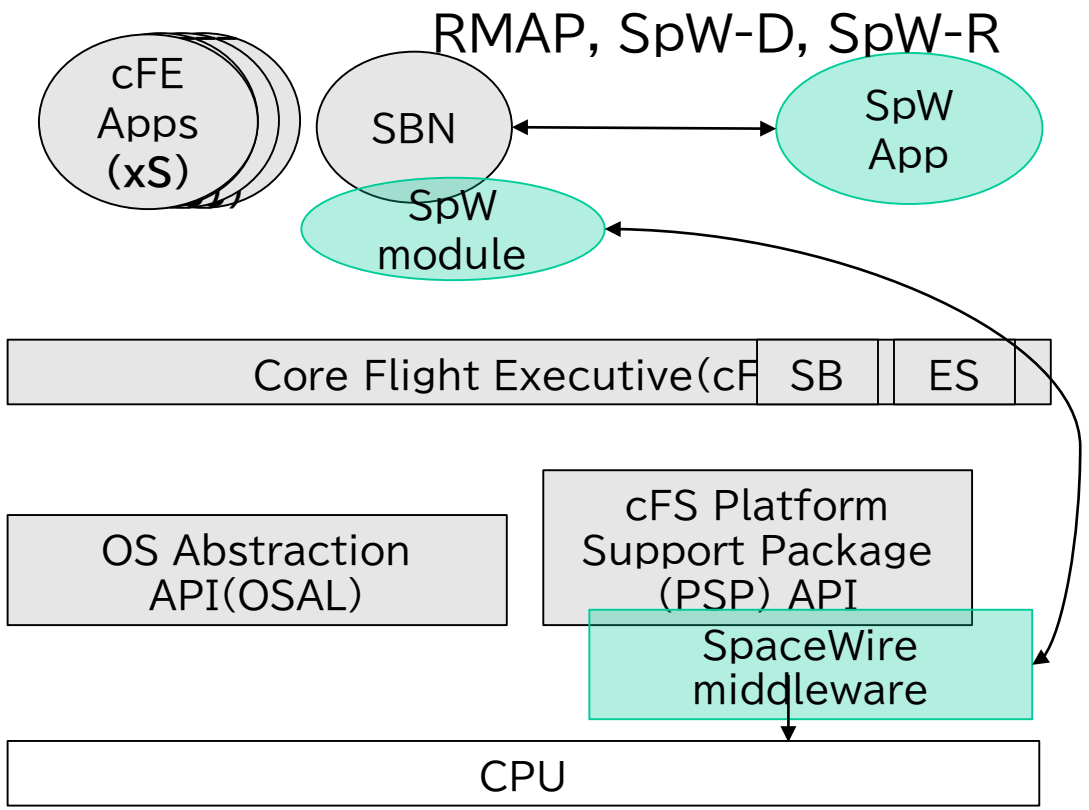
- OSALを入れ替えることで様々なOSに対応可能
  - Linux(posix),VxWorks,Rtems,TOPPERS/HRP2
- 地上系も含めたアプリケーションがLinux上での検証が可能

cFE/CFSのアプリケーション通信のしくみ

- SB(Software Bus)
  - ソフトウェアモジュール間のメッセージ通信の仕組み
  - メッセージはCCSDSパケット想定(変更も可能)
- SBN(Software Bus Network)
  - 通信プロトコルを用いた、(搭載機器間)ノード間通信のインタフェース
  - Pub/Subによるノード情報の交換が可能
  - ハウスキーピング機能
  - 下位レイヤの通信プロトコルはUDP,TCPだけでなく、シリアルやSpaceWire、DTNなども参考実装があった



SB/SBNへのSpaceWireの対応



- SpW AppはSBNのIFを使用
- SpW　moduleを作成
- SpWミドルウェアをターゲット向けに移植
- 対向デバイスはSpaceWireのターゲットデバイスを使用

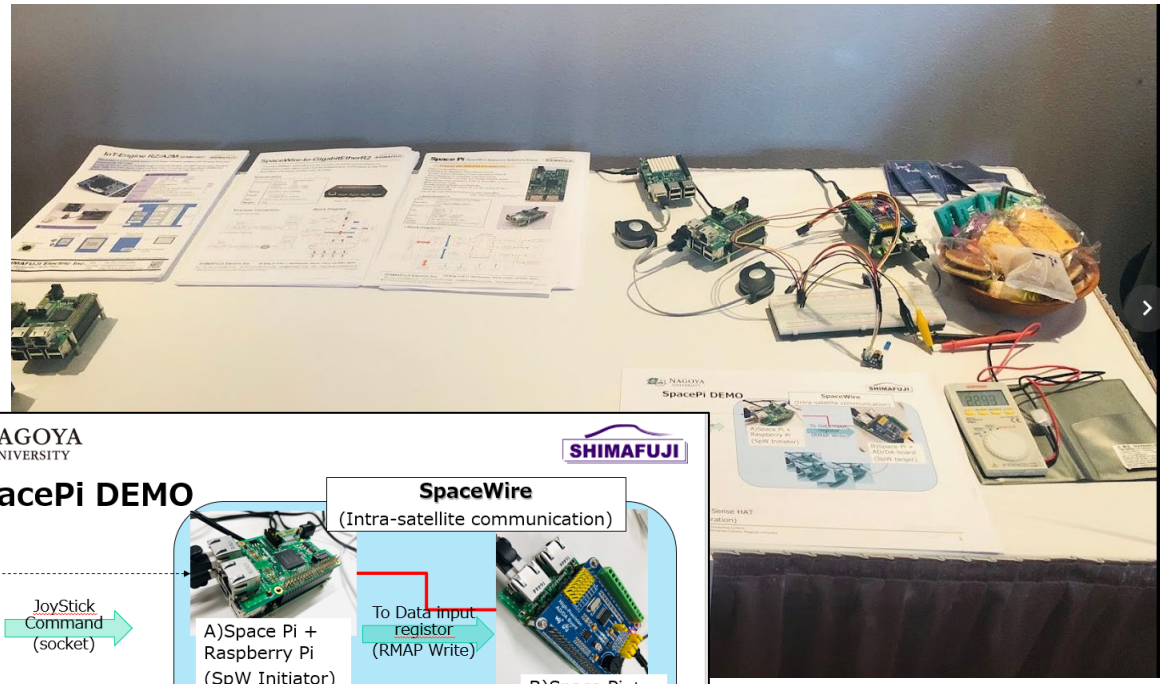
民生機器を用いたSpaceWire環境の構築

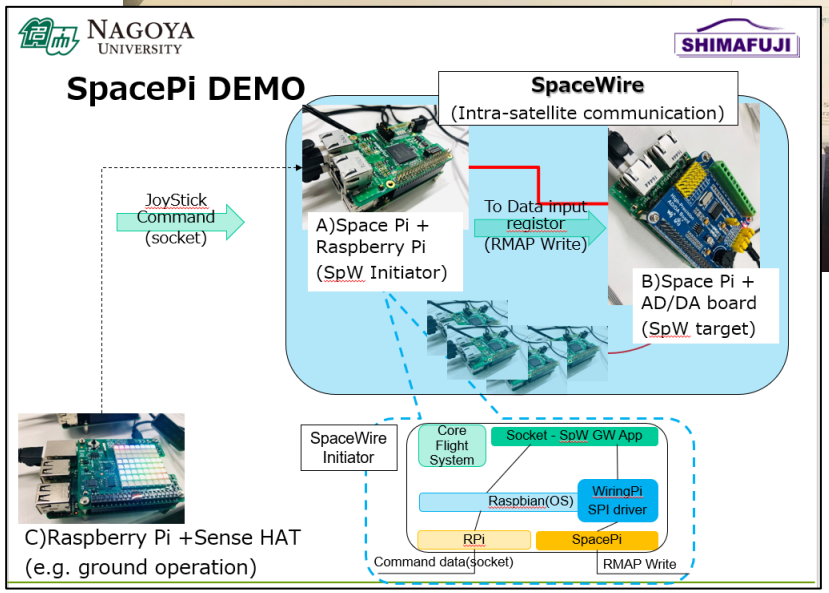
- SpaceWireを使った安価な環境の構築
  - シマフジ電機社製のSpacePi
  - RJ-45コネクタを用いて2SpaceWireポートを実現
  - RaspberryPi3の拡張ボードとして動作(SpW イニシエータ時)
  - 単体でSpWターゲットとしても利用可能

FPGA	Spartan-6（XC6SLX16-2FTG256I）
SpaceWire	2 Port （RJ-45） Link Speed：50Mbps（MAX）
Raspberry Pi I/F	SPI
Size	65（W）mm × 56（D）mm
Power	+5V （Supply from Raspberry Pi or AC adapter） Option : AC adapter

SmallSatConferenece2019(@USA)でのデモ

- JAPANブース内でSpacePiを使ったデモを実演

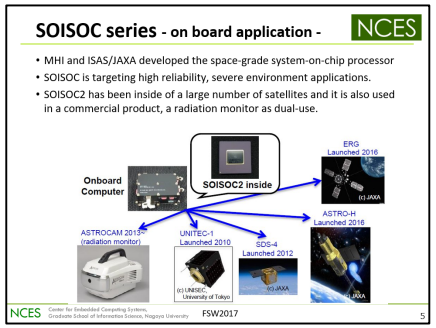




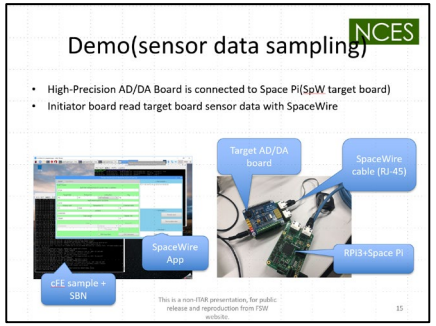
デモ機材とチラシ

搭載ソフトウェアの構築技術の成果

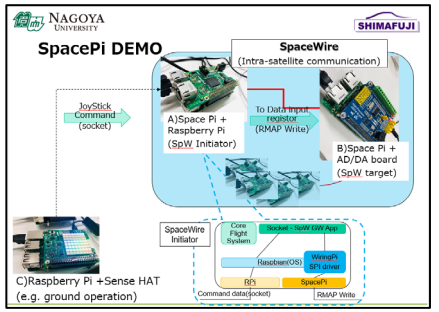
2017年度
- TOPPERS/HRP2のcFSへの移植(FSW2017)
  - MHI製耐放射線マイコンSOI-SOC3へのSpaceWireOS対応



2018年度
- cFS(SBN)のSpaceWire対応(FSW2018)
  - SpacePiを使ってSBNでのSpaceWireプロトコルの対応
  - AUTOSARとの機能比較(ETNET2019)



2019年度
- SpacePiのcFSへの移植およびデモ(SmalSat2019)
- SpaceWireRMAPライブラリのSpacePi対応



AUTOSARとcFE/CFSとの機能比較

要件項目	cFS	AUTOSAR CP	AUTOSAR AP
アプリIF層	SB	RTE	ARA
フォーマット	CCSDS(変更可能)	Signal(規定なし)	規定なし
通信方式	・Telemetry Send ・Command Send	・Sender/Receiver ・Client/Server	・Event ・Method ・Fields
データ方向	**単方向**	**単方向(SR) 双方向(CS)**	**単方向(Event) 双方向(Method)**
同期通信	**非同期**	**同期、非同期**	**同期、非同期**
トポロジ	1:1, 1:n, m:1	0:1, 1:1, 1:n, m:1, n:m	CPと同じ
接続方式	**Pub/Subによる動的設定**	**ビルド時の静的設定のみ**	**・静的設定 ・計画された動的設定**
通信プロトコル	TCP, UDP, DTN, SpaceWire	PDU, SOME/IP	TCP, UDP, SOME/IP, DDS
通信物理層	Ethernet, Wireless, Serial, SpaceWire	CAN, LIN, FlexRay, Ethernet	Ethernet

宇宙機ソフトウェアもプラットフォーム化の流れではあるが…

ETNET2019発表資料より

今後の取り組み

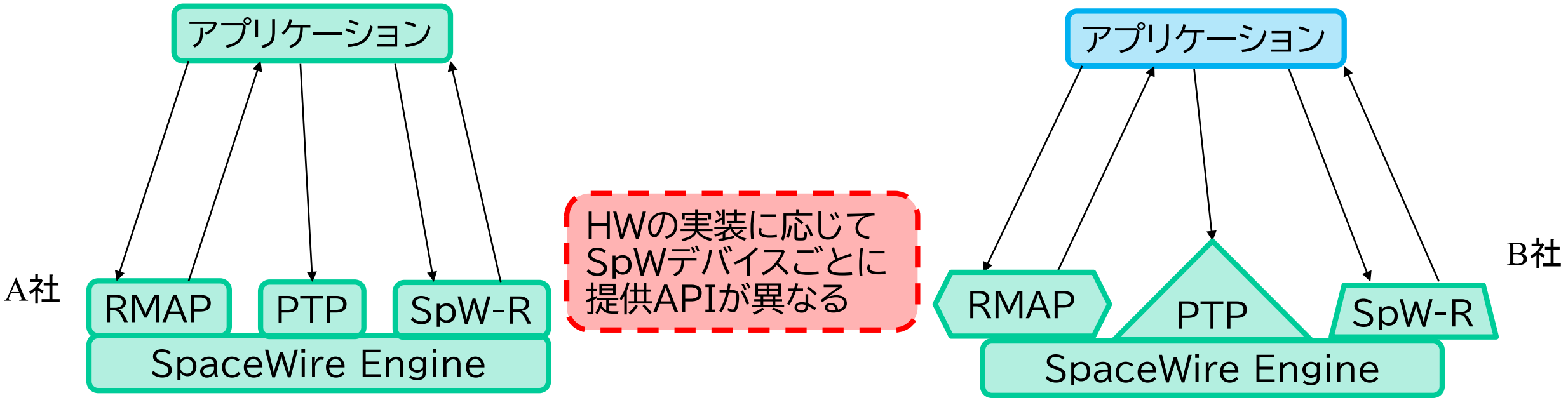
- 衛星搭載ソフトウェアも標準プラットフォーム化には進むが…
  - 中大型衛星だとミッション部がどうしても単体開発になりがち
  - プロジェクト単位の開発だとアプリケーションの再利用ケースが少ない

- かみ合わせテスト時に通信アプリケーションの不具合を検出
  →搭載機器単体テストではOKだが機器のかみ合わせで問題が発覚
  - ネットワーク設計情報が正しく伝わっていないケース
  - シミュレータでのアプリがそのまま搭載機器で使用できない
  - 採用デバイス(メーカ)ごとにアプリケーションIFが異なり情報共有が難しい

デバイスに依存しない
SpaceWireの標準インタフェースの検討

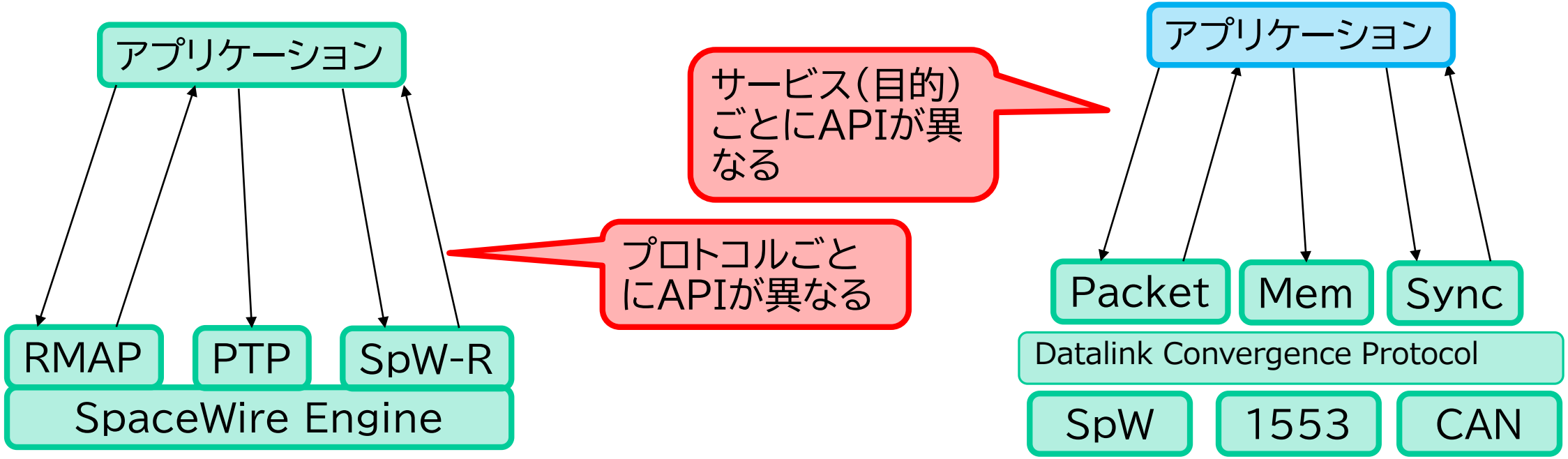
検討するレイヤと現状の課題(1)

1. デバイスで提供しているAPIが異なるため、同じ機能のアプリケーションなのにAPIが変わってしまう(アプリケーションの作り方も変わる場合がある)
   - 複数社の場合(各社ユーザニーズに合わせたAPI(SDK)の提供を行っている)
   - 例1:異なる会社でのアプリケーション作成の場合
   - 例2:同一アプリケーションを実機とシミュレータで利用する場合



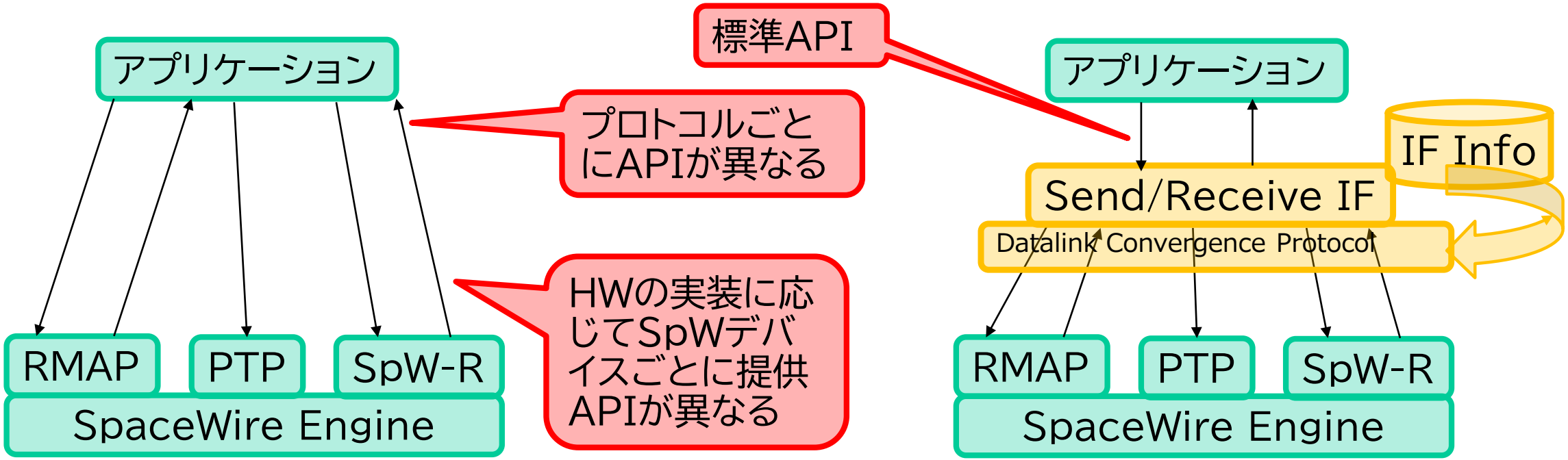
検討するレイヤと現状の課題(2)

2. 各社SpWプロトコルごとにAPIが提供されているが、SOISではサービスごとに応じたインタフェースが規定されている
   - SOISではプロトコルを変更した場合でもアプリケーション側のインタフェースは変更しないことが望まれている

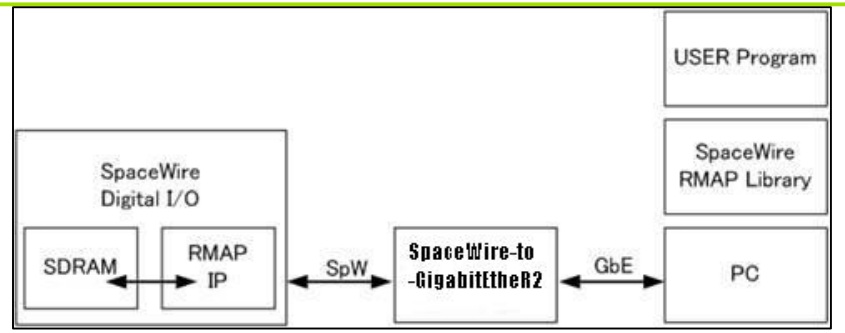


検討するレイヤと現状の課題解決案

- Socket通信でのSend/Receive IFのようなAPIを提供する
  - IF毎に情報(IF info)を持ち、プロトコルレイヤ(デバイスごと)のパラメータを登録することができるようにする
  - 新しいデバイスやプロトコルができた場合は、IF Infoへの登録パラメータの生成・更新だけでアプリケーションのインタフェースは変えなくてよい



SpaceWireの抽象化されたインタフェースの実装検討



言語による実装

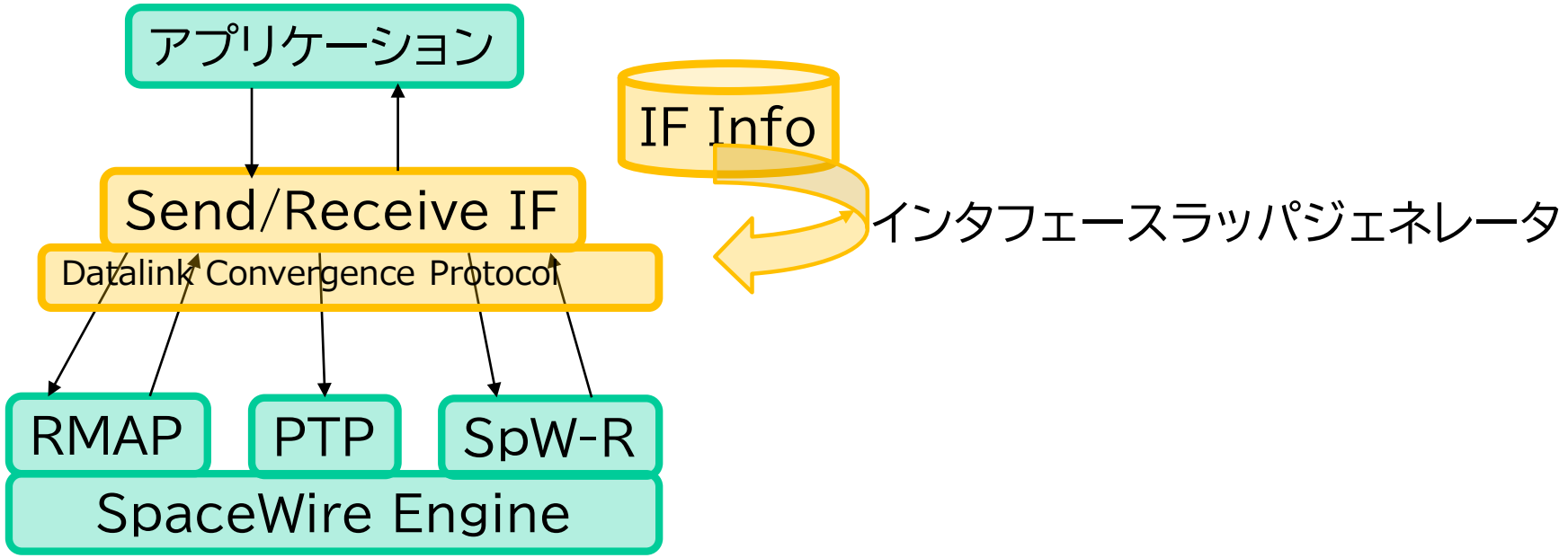
- ポリモーフィズムを持つ言語で実装を行う(C++)
- SpaceWireRMAPライブラリの利用
  - C++実装によるSpaceWireRMAPのクラスライブラリ
  - https://github.com/yuasatakayuki/SpaceWireRMAPLibrary
  - SDS-1やASTRO-Hなどの衛星開発・テストで利用
  - SpaceWire-to-GigabitEther(ルータ)向けとして使用

- SpaceWireRMAPライブラリの拡張(SpacePiへの対応)
  - SpacePiクラスとドライバの開発
  - SpaceWire通信、RMAP通信での確認を実施

SpaceWireの抽象化されたインタフェースの実装検討

インタフェースラッパによる実装

- C言語での実装ではプロトコルごとに対応するインタフェースラッパを用意する
- インタフェースラッパのジェネレータの開発(予定)
- ジェネレータ実行時のパラメータ(IF Info)についても検討



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: https://www.nces.i.nagoya-u.ac.jp/
email: nces-office@nces.i.nagoya-u.ac.jp