第23回TOPPERS開発者会議開催レポート

2022年10月23日（日）、24（月）に、オンラインにて、第23回TOPPERS開発者会議を開催しました。TOPPERS開発者会議は、TOPPERSプロジェクトの開発成果物の開発者・利用者が集まり、合宿形式で集中的に議論、開発する会議です。非会員も参加可能です。今年もコロナ禍のためオンライン開催となりましたが、非会員も含め多数の方々が参加され、活発な議論が展開されました。本レポートでは、その魅力をたっぷりお伝えします。

ゲストトーク: OSS開発時代の自動車業界におけるSBOM導入と運用

今年のゲストトークは、近年OSSのセキュリティやライセンス管理に関連して注目を集めているSBOM（Software Bill of Materials、ソフトウェア部品表）技術について、Covalent 株式会社の小林 弘樹 様、森 恭一 様のお二方に「OSS開発時代の自動車業界におけるSBOM導入と運用」と題して講演を頂きました。

講演の前半は、小林様より SBOM の技術的概要とその必要性に関して説明して頂きました。大規模化する車載ソフトウェア開発において各種オープンソースソフトウェアを活用することは必須になっていますが、その一方でOSSライセンスへの準拠、セキュリティ脆弱性管理などのリスクが問題となっています。またソフトウェア製品の大規模化、複雑化のため製品に含まれる全てのOSSを把握すること自体が難しくなってきています。これらの解決手段として SBOM技術が注目されています。SBOMとは対象ソフトウェアに含まれるOSS製品の一覧を表すデータであり、手作業で作成する事も可能ですが、対象ソフトウェアのソースコードやバイナリファイルを解析しSBOMを自動生成する支援ツールが様々なツールベンダから提供されています。これらを活用することで、意図しないOSSの混入を防止しライセンス違反やセキュリティ脆弱性のリスクを回避することが出来ます。SBOMのデータ形式に関しては幾つかの方式が提案されており、代表的なものとしてSPDX（ISO/IEC 5962:2021）などがあります。現在、様々な産業界において SBOM 利用を推奨するための法整備が進んでおり、一部の国際標準や政府調達案件ではSBOMを義務づける動きもあります。車載分野では、2021年1月の道路運送車両の保安基準改正により一般車両におけるサイバーセキュリティ等に関する規則 UN-R155/156 の適用が順次義務化され、SBOMによる管理を含めたセキュリティ対策が必須となりつつあります。

後半では、2021年に経済産業省が行ったSBOM実証試験に参加し試験運用の実務を担当された森様から、実証試験で得られた知見等に基づき、SBOM導入、運用についての課題やノウハウについて紹介して頂きました。現場へのSBOM導入を円滑に進めるには、自社で必要となるOSS管理の要件を明確にした上で、PoCにより要件を充足できる SBOMツールを選定し、運用体制の構築を進めることが必要です。また、経済産業省の実証実験の結果を通じて得られた知見としては、SBOM運用業務の中では「影響範囲特定」の作業負荷が特に大きく、他の業務へ与える影響も大きかったとのことでした。使用したと認識がないOSSがSBOMツールの走査で発見された場合、誤検知であるか否かの確認作業で一件あたり80分以上の工数が発生しました。また、脆弱性が報告さ

れたOSSのアップデート作業を想定した工数試算でも、アップデートに伴う影響範囲の分析が全工数の2割を占める結果となりました。これらの作業を効率化するにはトレーサビリティ管理が重要であり、Covalent株式会社ではZIPC TERASなどのトレーサビリティ管理ツールとの連携にも取り組んでいるそうです。また、SBOMツールでOSS使用が検出された後の対応だけではなく、開発プロセスの上流工程から適切なOSS利用を検討するために適切な「OSSポリシ・ガイドライン」を策定するなどの取組みも重要とのことでした。

講演後の質疑では、TOPPERSとしての SBOM 活用について、様々な観点から活発な議論が行われました。

高田会長からは、ソフトウェア技術者がOSS を利用した際に適切に記録を残す事が望ましく、SBOM手動作成を支援するツールが欲しい、との課題提起がありました。これに対しては、パッケージ管理ツール(rubygem、npmなど)との連携などの技術提案があった一方、手作業による記録漏れや故意の隠蔽を防止する観点から自動検出を積極的に使うべきとの意見もありました。

TOPPERSが開発する各種OSS製品が各社のSBOMツールのデータベースに適切に登録された状態とするにはどうすればよいか、という点も議論となりました。講師の先生方からは、各ベンダに対して未登録OSSの登録を依頼するとほぼ即座に登録が行われる、CVE等の公の脆弱性データベースは主要ベンダがチェックしているのでこれらへの情報登録を適切に行うのがよい、といったアドバイスを頂きました。

最後に、TOPPERS としての今後のSBOMへの取組み方針については、残念ながら具体的な結論には至りませんでしたが、高田会長からはTOPPERSリリース成果物に対して SPDX 形式のSBOM情報を添付するところから始めてはどうか、と提案がありました。TOPPERS会員でSBOM に興味のある方は、率先してチャレンジしてみるのはどうでしょうか。

議論1: SPIKE-RT 議論・ハンズオン

朱 義文 様、松原 豊 様(名古屋大学)

教育現場を中心に人気のLEGO SPIKE Prime上で動作するTOPPERS環境として開発が進められているSPIKE-RT について、解説、ハンズオン、議論を行いました。

最初に、朱 義文様から、SPIKE-RT(LEGO SPIKE Prime 向けのRTOSベースのソフトウェアプラットフォーム(SPF))の概要についての説明がありました。具体的な項目としては、

- SPIKE Prime Hub(マイコン)の詳細
- Pybricks(LEGOのコンピュータ向けOSSのSPF)について
- SPIKE-RTの構成(Pybricksの軽量OS Contiki をASP3上に移植)
- デバイスの対応状況

などについて、話をしていただきました。

つづいて、SPIKE-RTを用いてアプリケーションを開発するための一連の作業について、下記のgithubに置かれている解説とソースコードをもとに、ハンズオンを行なっていただきました。
https://github.com/spike-rt/spike-rt/

ビルドやテストなど様々な解説のドキュメントは、ここにあります。
https://github.com/spike-rt/spike-rt/blob/main/docs/ja/

最後に、今後の発展や方向性などについて、名古屋大学の 松原 様を中心に、議論を行いました。大きな項目としては、ロボコンへの協力の具体的な計画、サンプルアプリケーション、教材開発の3つについて、議論や提案が行われました。

議論2: TOPPERSカーネル仕様に関する最近の話題・課題

高田 広章様（名古屋大学）

TOPPERS第3世代カーネル（ITRON系）仕様に関して，最近話題・課題になって いることについて説明し，議論するのを目的とされ、**具体的には，優先度上限ミューテックスの仕様を深堀りした内容になりました。**

1 TOPPERS第3世代カーネル統合仕様の最近の変更点
仕様のバージョン間の差分は、仕様書の付録Aにリストアップされているとのことでした。
発表内容は仕様書として公開される前の状態のものであり、TOPPERSプロジェクト会員はSubversionのリポジトリで確認することができるそうです。

1.1 sus_tskとter_tskの呼び出し許可方法の拡張
sus_tsk(タスクをサスペンドする)、ter_tsk(タスクを強制終了する)は無制限に呼び出すと危ないシステムコールであり、他のパーティションで排他制御しているミューテックスを取得中の自パーティションのタスクをサスペンドさせたり、強制終了させれば、他のパーティションをとめてしまう危険性があるため、sus_tsk、ter_tskの呼び出し制限を細かくかけれらるように仕様変更したとのことでした。

注：パーティショニング：UNIXなどでMMUやCPUの保護機構を用いて実現しているプロセス間の分離・保護に類似した仕組み。タスクなどカーネル管理資源をグループ化し、グループ間での分離・保護を行う

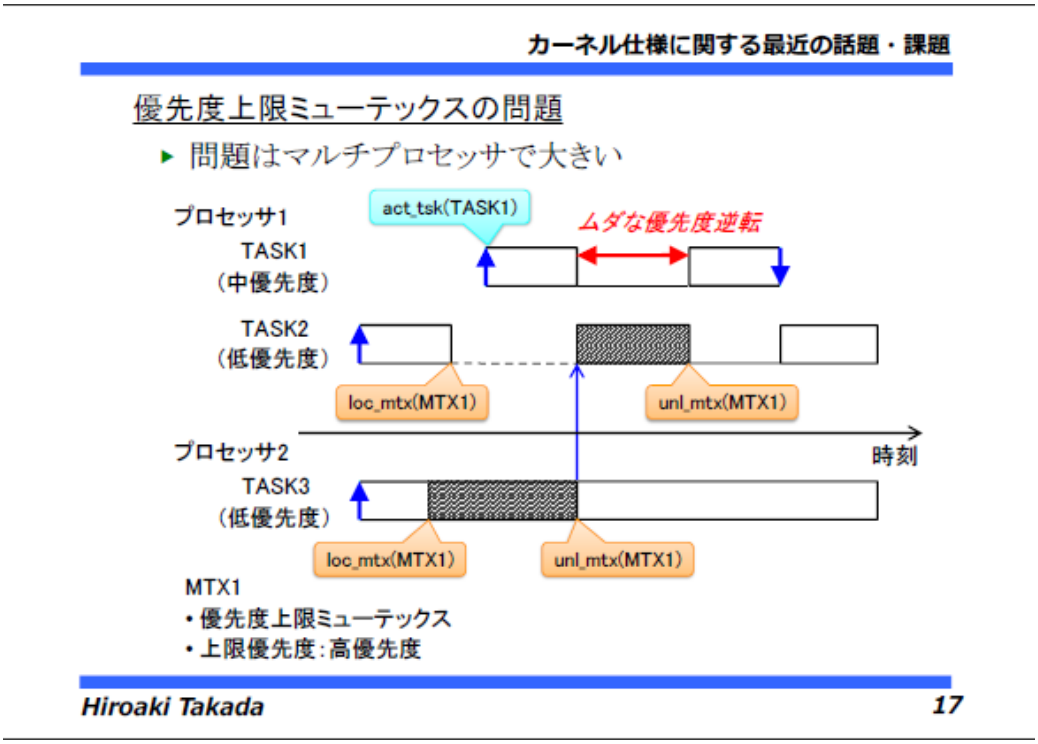
1.2 優先度継承ミューテックスの導入
μITRON4.0仕様に定義されていたが、TOPPERS新世代カーネル仕様で削除し、第3世代カーネル仕様でも存在しなかったが、ユーザからの要望が多かったため、復活させたとのことでした。
同優先度のタスク間での優先順位の扱い、第3世代カーネル仕様で導入されたサブ優先度との関係で、仕様検討が必要だったそうです。
(Subversionリポジトリのtemporary/mutex_spec.txtを参照)
この仕様変更をもとに、ASPカーネルに優先度継承パッケージを追加し、次のリリースでは出したいと言われました。

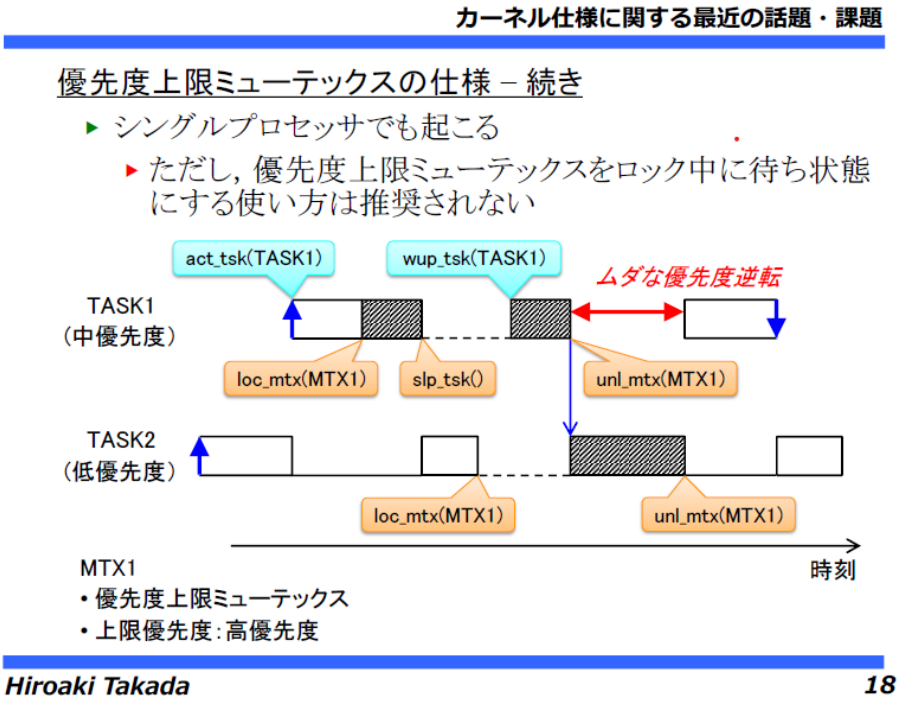
2 ASPカーネル機能拡張・チューニングガイドの紹介
これは、ASP3のパッケージの中のasp3/doc/extension.txtのことで、ASP3カーネルを改造して、機能拡張や性能改善を行う方法（またはヒント）を説明したドキュメントだそうです。
カーネル仕様に対して、様々な要望（機能拡張等）があるが、標準パッケージに取り込むと、使わないユーザにとってはオーバーヘッドになるため、標準的ではない要望に対する2つのアプローチを用意しているとのことです。
・拡張パッケージを容易
・改造方法を示す
（ただし、改造方法に対応した網羅されたテストは行っていないそうです）

3 TOPPERS第3世代カーネル統合仕様の課題
3.1 マルチプロセッサにおけるメッセージバッファの使用制限
TOPPERS開発者会議2020で紹介し、仕様変更の方向性はほぼ決めた状態だが、実装量が多いため、まだ実装していないとのことでした。
3.2 優先度上限ミューテックスの問題
これは、この後改めて深堀りするとのことでした。

4 優先度上限ミューテックスの問題とテスト実装
4.1 ミューテックスの仕様
タスクの現在優先度
タスクの現在優先度は、常に、以下の優先度の中で最も高い優先度に設定される。
そのタスクのベース優先度
そのタスクがロックしている優先度上限ミューテックスの優先度上限
loc_mtx
対象ミューテックスがロックされていない場合には、自タスクによってロックされている状態になる。
対象ミューテックスが自タスク以外のタスクによってロックされている場合には、自タスクはミューテックスのロック待ち状態となり、対象ミューテックスの待ち行列につながれる。
unl_mtx
対象ミューテックスの待ち行列にタスクが存在する場合には、待ち行列の先頭のタスクが待ち解除される。
対象ミューテックスは、待ち解除されたタスクによってロックされている状態になる。
待ち解除されたタスクには、待ち状態となったサービスコールからE_OKが返る。
待ち行列にタスクが存在しない場合には、対象ミューテックスはロックされていない状態になる。

4.2 問題点





（発表資料「カーネル仕様に関する最近の話題・議題」（2022年10月24日　高田広章）より引用）

4.3 仕様の変更案
unl_cpuでは、待ち状態のタスクにミューテックスをロックさせず、単に待ち解除だけを行い、次にそのタスクがスケジュールされた時にロックさせる。待ち状態のタスクをすべて待ち解除することが必要。

4.4 変更案の実装
loc_mtx()
ミューテックスをロックしていたタスクによるunl_mtx()呼び出しにより、ロック待ちが解除されたことを示すエラーコード（カーネル内でのみ使用）E_RETRYが返ってきたら、loc_mtx()を抜けるのではなく、loc_mtx()内の処理を再実行する。
タイムアウトを示すエラーコードの場合はループを抜ける。

従来の仕様では、自タスクがロック待ちを解除されるのは、ミューテックスをロックできたか、タイムアウトしたことを意味していた。
変更案では、これら以外に、loc_mtx()を抜けることなく、loc_mtx()内の処理を再実行する状態が追加される。

mutex_release(ミューテックスのロック解除処理)は処理内容を入れ替える必要がある。
・ロックしているタスクが存在しないことにする。
・待ち行列につながれタスクを、待ち行列の先頭のタスクから順に全て町解除する。
・E_RETRYの定義追加
・ミューテックスの待ち行列にタスクを挿入する処理の変更

4.5 この仕様・実装の課題
・タイムアウトの振る舞い
tloc_mtxのタイムアウトは1回ずつの待ち状態の継続時間に適用され、ミューテックスのロック待ち全体の時間には適用されない。
・強制的な待ち解除(rel_wai)の振舞い
rel_waiで確実に強制待ち解除できるとは限らなくなる
・ベース優先度の変更(chg_pri)のE_ILUSEエラー
これらを問題なしと考えることもできるが、ユーザには使いにくいと思われる。

4.6 仕様の修正案(2)(追加仕様)
「ミューテックスロック再実行中」の概念の導入
タスクがミューテックスのロック待ち状態から待ち解除された後、ロック処理を再実行するまでの間の状態。
この状態の場合の処理を追加し、従来の仕様での振る舞いとの整合性を持たせる。

4.7 仕様の修正案(2)の実装
最初の修正案に対して、50行ぐらい

4.8 TOPPERS第3世代カーネルに取り込むか？
比較的小さい改造で、優先度上限ミューテックスの問題点を改善できる。
しかし、これまでの仕様と大きくコンセプトが異なるため、仕様を全面変更するのは、インパクトが大きい。
従来の仕様をサポートしつつ、新しい振舞いのミューテックスを新規に導入するアプローチが有力。

参加者とのディスカッション（順不同でピックアップした意見。問は参加者、答は高田先生）
・「優先度上限ミューテックスの問題」で指摘されていることは、気持ち悪いは思うが、（発生するのは）仕方がないとも思う。割り切る問題ではないか。

・優先度逆転している期間が短ければ問題なく、長ければ他のプロセッサにマイグレーションすればよいのでは。
・「優先度上限ミューテックスの問題」で指摘されているケースは、ワーストケースとしては、あの条件の時は起こることを覚悟していないといけないのではないか。
・マイグレーションを考慮すると、解が広がる。
(問)「優先度上限ミューテックスの問題」は、コアに対するタスクの割り付けが固定の場合固有の問題か。
(答)マイグレーションのアルゴリズム次第だと思う。もしLinuxのミューテックスがこういう実装をしているなら、Linuxにもこういう問題があると思う。
・優先度上限でデッドロックを防げるのは、シングルプロセッサのみで、マルチプロセッサでは出来ない。
・優先度上限を使いたいという要望は、シングルプロセッサでの経験から(今回議論したような細かいことを考慮せずにに)来ているのではないか。
・優先度上限自体が、シングルコアをベースにしたコンセプトだろう。
・優先度上限をマルチコアに持ってくること自体が合わないことで、サポートしないというのも有りではないか。
・マルチコアで優先度上限を利用したいのは、「コア間のロックをなるべく早く解除したい。」というモチベーションがある。
・コア間のロックが存在すると、並列度が下がる。
(問)実装のテストはどれぐらいで十分か
(答)難しい。カバレッジによるテストはやるとしても、「パスが抜けている」という問題に対しては、検出できない。
直接の答えではないが、やったことの紹介をする。
(画面共有で「TOPPERS/ASPカーネル優先度継承ミューテックスのテスト」(test_inherit.txt)を映す)
A)2つの観点の網羅度を考察
(1)統合仕様書のタグの網羅度(ブラックボックステスト)
(2)ソースコードの網羅度(ホワイトボックステスト)C1 100%
B)タグ毎にテスト振り分け
仕様を手続きにではなく、満たすべき性質として記述した場合は、テストが大変。
仕様から、テストすべき要素、条件、処理を洗い出し、すべてテストしなければならない。
ミューテックスのテストが最も複雑な場合だった。
C)テスト対象の処理が、どのAPIにあるかを数え上げ、それぞれのケース毎にテスト設計をする。
・テストの事前条件、事後条件も含めると、例えば10個のテスト項目をテストするのに、テストシーケンス百数十個になった。
D)ソースコードカバレッジの確認
カバレッジ計測ツール(gcov)を用いて、通っていないパスを通るテストを追加した。
・統合仕様書は外部仕様書で、その上位に要求仕様書が存在すべきで、要求仕様書はInvaliantで書かれるべき。要求仕様書と統合仕様書の一貫性の数理的な検証まで出来ていれば完璧。
・セマフォやミューテックスみたいなものはInvariantで書きやすいが、イベントフラグみたいものはInvariantで多分書きにくい。
・私(高田)としては、設計ドキュメントとか、テストで何をやったかは徐々にはなるかもしれませんが、公開していくつもりです。

議論3: TINET の検証に関する議論

議論3では、TOPPERS プロジェクトの組込み向け TCP/IP スタック実装である TINET をに関する脆弱性検証の活動について、二名からそれぞれ報告を行いました。

前半は、TOPPERS ホームネットワークWGの長島さん(㈱コアーズ)が「TINETのFuzzing環境構築とスレッドサニタイザ」について報告しました。ホームネットワークWGでは、ファジングテスト用ライブラリ libFuzzer を用いた TINET の検証作業に取り組んでいます。ファジングは大量のメモリを使用し処理速度も必要なので組込み機器

上ではなく Windows PC 上でテストを実行しており、Windows 上での動作のためにタスク切り替え機能をsetjmp/longjmpを用いて再現するなどの工夫を行っていることが報告されました。

Ethernet フレーム受信部を出発点として行ったファジングテストでは、ヘッダ内のサイズ値に不整合があった場合やARPテーブルが全て埋まった状態でARP応答を受信した場合など複数のケースにおいて不正なメモリアクセスの欠陥を発見するなど、検証活動としての実績が実際に得られています。ファジングで発見されたTINETの不具合とその修正については、github リポジトリ https://github.com/toppers/TINET 上において順次公開されています。

現在は、スレッド間のアクセス競合を検出するスレッドサニタイザ機能の適用に挑戦中とのことです。スレッドサニタイザはWindows 環境での動作に対応していないため、現在開発中のASP3カーネル POSIX ターゲット依存部の実装を用いて Linux 上での実行を試みています。しかし、現状ではまだうまく動作していないとのことで、POSIXターゲットの内部設計に関して開発者である高田会長を交えた技術検討も行われました。

後半は、開発者会議実行委員会の堀（個人会員）から、「OSS静的解析ツールによるTINET検証」について報告と議論を行いました。自動車や航空宇宙分野で用いられている商用の静的解析ツールは高性能ですが高価であるため、無償で利用可能なOSS静的解析ツールである infer (Facebook research)などを用いて検証作業を行っています。TINETのTCP, UDP 実装コードに対するこれまでの適用試験では、想定よりも欠陥検出の件数が少なく、またファジングで発見された既知の欠陥についても適切に検出できていない、という結果となりました。まだ予備試験段階なので、ツール適用方法の見直しなども含めて更なる検討を進める、との事でした。また報告後の質疑では、参加者の方々の開発業務における各種静的解析ツール製品の活用状況などについて、多くの有益な情報提供を頂くことができました。

議論4: TECS WGの最新研究

(1)TOPPERS/HRMP3のTECS対応とマルチコアの有効活用に向けて
高相 義忠 様（埼玉大学）

TECSを用いて、保護機能対応マルチプロセッサリアルタイムOS「TOPPERS/HRMP3」の以下の課題を解決することが目的だそうです。
・対象システムの構造や設計を完全に理解する必要があり、手が出しにくく、再利用性・拡張性が低い
・システムごとの静的API - 既存の記述をそのまま再利用することが厳しい
TECSはTOPPERSプロジェクトで開発されている組込みシステムに適したコンポーネントシステムです。
マルチプロセッサに関わる「クラス」は、「マルチプロセッサに対応するためのカーネルオブジェクトの集合」であり、カーネルオブジェクトをグループ化して、グループ毎に各プロセッサに割付けます。
保護機能に関わる「ドメイン」は、「メモリ保護の単位であり、メモリ領域を区分けして、区画ごとに保護を施すか否かを設定」します。許可がないと非保護領域から保護ドメインにあるカーネルオブジェクトにアクセスできません。
提案された解決策は、HRMP3でのクラス・ドメインをTECSのリージョンと結びつけてCDLで記述するというものでした。
この場合2つの観点でクラスを決めます。
1.実行プロセッサによって分ける
2.実行開始後にプロセッサからカーネルオブジェクトを移動できるか否か（マイグレートしていいかどうか）で決める

ドメインは3種類(KERNEL_DOMAIN、なし、USER_DOMAIN)を用います。
TECSのCDLでリージョンごとにTECSのコンポーネント定義、組み上げ記述を書くと、それに基づいて静的APIが生成されるとのことでした。
現在の実装の進捗は、HRMP3のTECS対応版の移植し、デバッグ中であり、エミュレータ上でサンプルプログラムの動作確認しているとのことでした。

(2)TECSへのPublish/Subscribe通信の導入と統合
佐々木 駿様（名古屋大学）

Publish/Subscribe通信とは、疎結合のPublisher(送信者）とSubscriber(受診者）が、Topicを介して非同期的なメッセージ通信を行うことであり、1対多／多対多の通信が可能です。SubscriberはPublisherが誰かを知る必要がないです。
研究の目的は以下の通りだそうです。
1. TECSでのPub/Sub通信の導入を容易にする(ロー～ミドルレンジのMCUでも独立して動作できる程度の軽量な実装)
2. TECSの細かい粒度での最適化(組上げ時・動作時には送信先が静的に決まっている状況も想定する=>TECSの恩恵を受けやすい)

既存の実装であるMQTT、DDS(Data Distribution Service)などは、サーバが必要であったり、実装が重かったりというデメリットがあるが、Eclipse Iceoryx(TM)というゼロコピーのプロセス間通信ミドルウェアは、求めているものにかなり近かったそうです。
提案は、TECSに適したPub/Subフレームワーク（複数の利用状況を想定し、それぞれに最適化レベルのPub/Sub実装を提供する）であり、内部をコンポーネント化し、組上げ時に最適なコンポーネントを選択できるようにされています。
また、以下の観点から切り分けたそうです。
・Publisher、Subscriberがアドレス空間を共有するか
・組上げ時点でTopic、Publisherノード、Subscriberノードが決まっているか

現時点でPublisher、Subscriberが同一アドレス空間になるケースが実装予定だそうです。
組上げ時点でTopic、Publisherノード、Subscriberノードが決まっている場合は、静的に構成できる部分が増えるため、組上げ時点で購読者リストが静的に決まり、使用RAM削減が期待できるとのことでした。
評価は内容がまだ未定であるが、比較対象はIceoryxなどを、また静的に配信先が決まるか否かがどれほど影響を与えるかを評価ポイントの一つとして考えいるそうです。
明確な仕様が定まっていない段階の課題には、データコピー、Subscriberが元のデータに直接アクセス、配信時のポリシーの選択肢(1度の送信のみ、再送／データ保持)、動作中に送信先が変わる可能性を挙げられていました。

参加者とのディスカッション（順不同でピックアップした意見)
・Pub/Sub通信で有用だと思うユースケースは通信の多重化
・車のCANはPub/Subの発想に近い。
・ブロードキャストで同報通信することは1対多ということができる
・1対1通信だと、ECUを1個追加すると、送信側も直さないといけない。Pub-Subスタイルだと、Subscriberが1個増えるだけなので、送信側を直さなくてよい。その辺は大きいメリット。
・同じコンピュータに乗っっている場合と、違うコンピュータに乗っている場合のどちらもある。
・同じコンピュータに乗っている場合は、共有メモリを使う。
・別のコンピュータに乗っている場合は、CANでもいいし、DDSでもいい。

・同じECUにある場合でも、別のECUにある場合でも、アプリを直さなくてよいとしたいのではないか。

(3)組込みコンポーネントシステム向き自動テストフレームワーク
富森 陽 様（埼玉大学）

組込みコンポーネントシステム開発におけるテストが困難(テスト対象が多い、テストが複数の段階、実機を用いたテスト)を挙げ、解決策として組込みコンポーネント用のCI環境構築を提案されました。
現在の状況は、既存研究であるスタブ未対応の単体フレームワークTECSUnitの半自動化したとのことです。
今後の課題は、スタブが必要なテストが行えない、結合テストに未対応、パソコンのみのテストと実機を用いたテストの差が未確認とのことでした。
具体的には以下に示す1,2までが出来ており、3.4.5を今後の予定だそうです。
1. TECSUnitをパソコンのみでビルドできるように修正
2. TECSUnitのビルドとその実行（テスト結果の表示）を自動化
3. スタブも対応するようにTECSUnitを修正
4. 結合テストを行えるように編集
5. エミュレータを用いてテスト

参加者とのディスカッション（順不同でピックアップした意見）
・Elixir_git_hooksの紹介
[qgadrian/elixir_git_hooks: 🪝 Add git hooks to Elixir projects](https://github.com/qgadrian/elixir_git_hooks)
・Elixirはテストコマンドを備えており、gitでpushする前にローカルでテストを実行し、パスしないとpushしないことが可能。
・Git自体がサブコマンド実行時にhookスクリプトを実行させて、その実行結果によりサブコマンドを実行する・実行しないを決める仕組みを持っており、Elixir_git_hooksはそれも利用しているのだろう。
・GitHub Actions 上のテストでは、TECSジェネレータの実行、コンパイル、テストプログラムの実行を行い、テストプログラムがテストケース毎の成功、失敗を報告し、テスト全体の成功、失敗を出力する。
・Github Actionsのステップでプログラムがエラーになれば、Github Actionsがそのワークフローが失敗だと認識するが、まだそれには対応していない。
・テストケースの管理も大変ではないか

議論5: OpenAMP(rpmsg)の仕様とTOPPERSでのサポート方針

はじめに、OpenAMPについての紹介がありました。

OpenAMPは異なるアーキテクチャのコアの間で通信するソフトウェアで、LinuxとDSP間の通信を行う目的で開発されたもの。Linuxが持つ膨大なソフトウェア資産と、リアルタイム性が求められる組込みソフトの資産を、それぞれを一つのデバイスに実装するため、共有メモリによる通信を行う仕組みを実現していると紹介がありました。

TOPPERSでは同様のソフトウェアとしてMDCOMを提供しているが、Linuxの更新による影響を受けやすく、大きな修正が必要になることがありメンテナンスの部分で問題があるそうです。その点OpenAMPはLinuxで標準でサポートされているため、メンテナンス性でメリットがあり移植を検討したとのこと。

続いて、ユビキタスAIの加藤吉之介様からLinux・RTOS間での利用について説明がありました。
OpenAMPにはFreeRTOSで利用するためのソースコードが提供されており、移植対象としてlibmetalという部分があるそうです。RTOSへの依存部が整理されており、FreeRTOSやNuttxの並びにTOPPERSを追加して対応し、RTOSに期待される関数を実装することで、移植が出来るようになっているとのこと。また、サンプルアプリとして

メッセージをエコーバックするものと、乗算結果を返すものがあり、FreeRTOSのAPIをTOPPERSのAPIに書き直すことで動作確認できたと紹介がありました。

次に、南山大学の本田晋也先生からRTOS・RTOS間で利用する方法についての方針の説明がありました。

OpenAMPのホスト側はFreeRTOSのみがサポートされているが、TOPPERSでも使えるように移植を検討しているとのこと。実装についてはこれからで、NXP社のi.MX RT685とST社のSTM32H7の実装について内部構造を調査されたようです。

i.MX RT685は、ARMとDSPの構成で、マルチコア通信用に、共有メモリ、割込み制御、排他制御を持っているようです。それを使ったキューでメッセージを送受信するもので、送信はAPI呼び出し、受信はコールバック関数が呼ばれる仕様となっているようです。また、静的オブジェクト生成に対応しているとのこと。

STM32H7はCortex-M7とM4の構成で、提供されるOpenAMPはNXPのものとはAPI名が異なっていたりと、派生したものとなっているようです。FreeRTOSのサンプルがあるが、同期のためのAPIを使っているわけではなく、受信はコールバックが使われているとのこと。ただTOPPERSに移植するにはAPI変換などの必要がないので、コールバックの方が移植しやすい見通しのようです。

質疑の時間では、参加者が使うためボードは購入可能かとの質問がありましたが、個人での購入には少し高額だとの話がありました。

また、OpenAMPを使った応用例として、ROSの通信に利用しRTOS側からLinux側へポイントクラウドを送信して処理するなどの検討をしていると紹介がありました。

2022年度開発者会議を振り返って

今年の開発者会議は合宿形式での開催を前提に準備してきましたが、新型コロナ感染状況が好転しなかったため、やむなくオンライン開催に変更しての開催となりました。直前での開催方式変更による混乱もありましたが、3回目のオンライン開催ということで Zoom や slack 上での議論も円滑にできたのではないか、と思います。

同じ「開発者会議」という名称であっても、比較的コアなTOPPERS会員が中心となる合宿形式と、非会員も含め幅広い参加者が集まるオンライン開催では、イベントとしての性格がかなり異なるのではないか、というのが、3年間オンライン開催を続けた時点での感想です。来年度の開催形態については、これからの開発者会議のあるべき姿を含めて検討したいと考えています。

NPO法人 TOPPERSプロジェクト
〒104-0042 東京都中央区入船1丁目5-11 弘報ビル5F
（一社）組込みシステム技術協会内
TEL & FAX: 03-6275-2981 Email: devconf@toppers.jp
Facebook : http://www.facebook.com/toppersproject