「実用化を目指した組込みシステム用
ディペンダブル・オペレーティングシステム」
平成 18 年度採択研究代表者

平成 21 年度
実績報告

徳田　英幸

慶應義塾大学環境情報学部・教授

マイクロユビキタスノード用ディペンダブル OS

§1. 研究実施の概要

マイクロユビキタスノード上に Linux OS をベースとしたディペンダビリティ機構を実現するために、以下の課題に取り組んだ。まず、無線の再接続時にも通信の継続を行う高信頼ネットワーク機構（研究課題 1）では、本機構の Linux 2.6 kernel での実装および評価を行った。来年度は本機構の P-Component 化を行うとともに、実際のアプリケーションを用いた実用性評価を行う。次に、バッテリ電力の予約に基づく電力管理を行う、高可用電力管理機構（研究課題 2）において、平成 21 年度は CPU、メモリ、ネットワークの使用量をプロセス毎に測定する機構（細粒度な電力消費量計測機構）の開発を行った。来年度は、複数の異なる計算機を用いてバッテリ毎の違いに対応するとともに、本測定機構を用いた電力予約の設計と実装を行う。最後に、オペレーティングシステムが提供するディペンダビリティの量と質を評価できるメトリクスの構築（研究課題 3）では、システムが必要とする、あるいはシステムを構成する機器に搭載される OS が提供するディペンダビリティ支援の量や質を議論するための基準を提案した。来年度はディペンダビリティ要求の評価とディペンダビリティ支援機構の評価を可能とし、その評価結果を定量化する手法、ならびに可視化する手法を検討する。

§2. 研究実施体制

（1）「徳田」グループ
①研究分担グループ長：徳田　英幸（慶應義塾大学、教授）
②研究項目
高可用電力管理機構、高信頼ネットワーク機構、ディペンダビリティメトリクス

§3. 研究実施内容

以下に、平成 21 年度の研究実施内容を課題ごとに述べる。

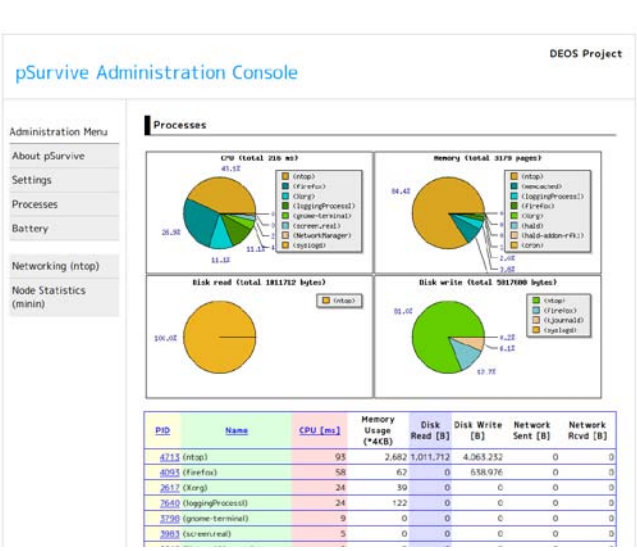
2-1. 高可用電力管理機構

高可用電力管理機構に関して、平成 21 年度は組み込み機器内で CPU、メモリ、ネットワークの使用量をプロセス毎に測定する電力使用量計測モジュールを構築した。昨年度時点ではネットワークについては機器全体としての使用量しか取得できなかったため、複数のプロセスがネットワークを使用する場合、プロセス毎のネットワーク使用量を計測することができなかった。組み込み機器の消費電力の内無線通信は大きなウェイトを占めるため、この点について早急に対応する必要があった。

そこで、本年度は新規に機能を拡張することで、プロセスとネットワーク使用量の紐付けを可能とするネットワーク計測モジュールを構築した。ネットワーク計測モジュールでは、従来 Linux のインターフェース情報から取っていたネットワーク使用量をパケットレベルの解析を行うように変更したことで、ポート番号別のネットワーク使用量を計測する。その後、ポート番号別のトラフィック情報をカーネルから取得したプロセスとソケットの対応情報と紐づけることで、プロセス毎のネットワーク使用量を計測している。

また、消費電力情報を時系列に記録する消費電力テーブルについて、従来の実装手法では専用のロギングプログラム上で管理していたが、稼働中のプロセスが増加したりログサイズが肥大化したりするにつれてシステムに大きな負荷をかけてしまうことが問題となっていた。そのため、より高速で読み書きが可能な memcached（http://memcached.org/）を採用し、高速化を図った。

これまでコマンドラインでのテキストベースのデータ取得しかできなかったものを、ブラウザからグラフィカルに閲覧する機能を追加した。これにより、従来は直感的に把握しづらかったリソース状況が把握できるようになった。具体的には、システムの中から特に消費電力の大きなプロセスを探し出したり、特定のプロセスについて過去のリソース消費量を時系列で閲覧することができるようになった。しかし、この表示方法についてはまだ完璧とは言えず、電池残量との関連は表せていない。今後電力予約を行っていく上でリソース消費量の可視化は重要な機能であるので、電力予約のユーザーインタフェースも考慮に入れ、開発を続けていく。



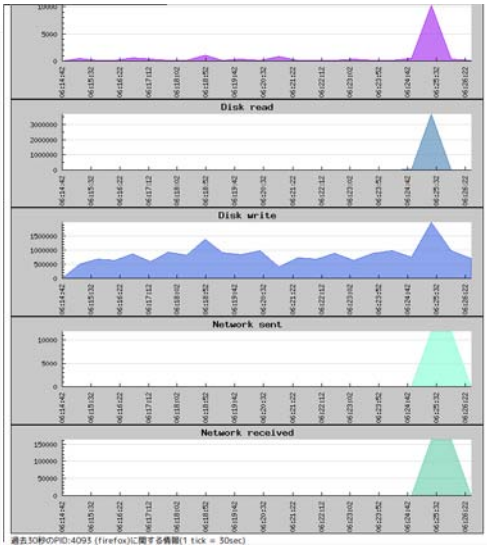


図1 全プロセスのリソース消費量の可視化　　**図 2 特定のプロセスの各リソース消費量履歴**

2-2. 高信頼ネットワーク機構

高信頼性ネットワーク機構の実現に対し、平成 21 年度は Linux 2.6 kernel において、Stream Control Transmission Protocol (SCTP)[RFC4960]を用いて異なるネットワークに端末が移動した際も通信を継続するハンドオーバ機能、および平成 20 年度に提案した高速ハンドオーバ手法[Honda'08] の実装および評価を行った。ハンドオーバ機能、高速ハンドオーバ手法ともに、FreeBSD と同様にシステムワイドのパラメータである sysctl の一つとして実装した。ハンドオーバ機能については、RFC5061[RFC5061] に準拠している。本実装を評価するために無線 LAN で構成された 3 つの IPv6 ネットワークセグメントによるテストベッドを用い、パフォーマンスの改善を計測した。以下にそのテストベッド環境を示す。

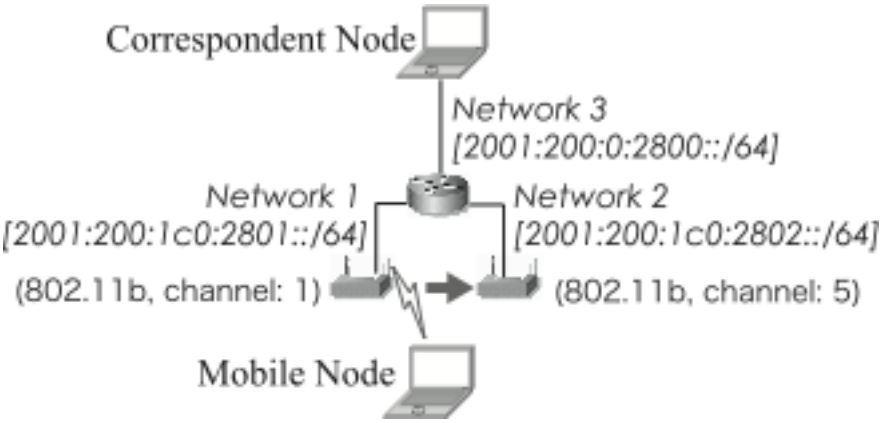


図 3: テストベッド環境

このテストベッドでは 3 つの IPv6 ネットワークを用意し、それぞれをモバイルノード (Mobile Node) の移動前ネットワーク (2001:200:1c0:2801::/64)、移動先ネットワーク (2001:200:1c0:2802::/64)、通信相手 (Correspondent Node) が接続しているネットワーク

(2001:200:1c0:2800::/64)として利用した。モバイルノードが接続するリンクは移動前、移動後ともに 802.11b を用い、通信相手が接続するリンクは 1000BASE-T の有線を利用した。無線 LAN を利用することにより、実験のリアリティを向上させた。本テストベッド上で、モバイルノードから通信相手にバルクデータ転送を行い、その間に Network 1 から Network 2 へのハンドオーバを行った。以下に実験結果として、ネットワーク移動時の物理的なコネクティビティ切断時間と実際の SCTP コネクションにおける通信切断時間の関係を示す。

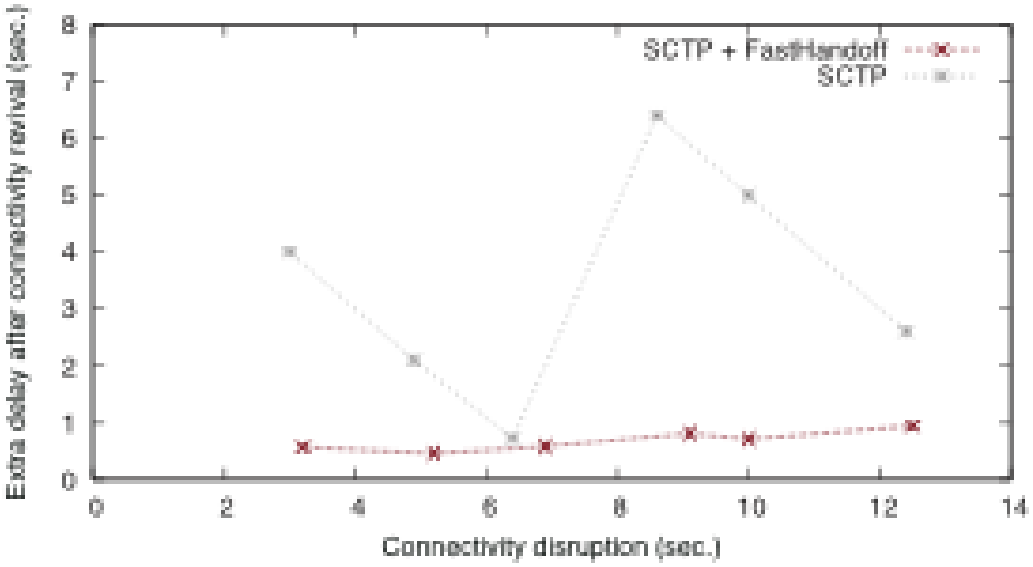


図 4: 切断時間と再接続時間

SCTP コネクションにおいて、通信切断後の遅延は以下の式（N はコネクティビティ切断中に再送タイムアウトが起こる回数、RTO はバックオフ前の再送タイムアウトまでの時間)で表されるため、ほぼ理想的な結果が得られた。

$$ExtraDelay = \sum_{n=0}^{N-1} RTO \times 2^n - DisconnectedPeriod$$

なお、ネットワーク切り替え後にコネクティビティが復旧した後に最大で 500 ms から 1000ms 程度の遅延が発生するが、これは SCTP スタック内のアドレスの追加及び削除の検知メカニズムの実装に起因するものである。ネットワーク層から SCTP へのアドレス通知は netlink と呼ばれるカーネル間通信機能を用いている。SCTP に関わらず、新たなアドレスは追加された時点で SCTP を含む他のモジュールに通知されるが、その際はまだ IPv6 アドレスは Duplicate Address Detection (DAD) 中であり、実際には使える状態になっていない。一方で、実際に使える状態になった時にその IPv6 アドレスを他のモジュールに通知するメカニズムは Linux kernel には存在しない。本実装では、SCTP スタック以外のソースコードに手を入れることを避けるため、DAD が終了してアドレスが使えるかどうかの判断は、タイマーを利用して SCTP スタックから 500 ms 毎にアドレスをチェックする実装にしている。そのため、実験結果でも最大で 1000 ms 程度のコネクテ

ィビティ復旧後の遅延が発生しているが、この部分に関しては容易にチューニングが可能である。今後の展望として、同時に大量のコネクションが存在する場合に、それらがコネクティビティ切断後の通信再開時に、バースト的に slow-start が開始され他のトラフィックに急激に変化をもたらす問題を解決予定である。

2-3. ディペンダビリティメトリクス

平成21年度は、システムが必要とする、あるいはシステムを構成する機器に搭載されるOSが提供するディペンダビリティ支援の量や質を議論するための基準(以降、ディペンダビリティ支援メトリクスあるいは単にメトリクスと呼ぶ。)を提案した。同メトリクスは、4種の定性評価項目と、それらをベンチマーク結果等の事実によりバックアップするD-Caseにより構成される。

阻害要因	支援の目的	フェーズ	対象機能
自然現象	可用性 (availability)	仕様	CPU
人的ミス	信頼性 (reliability)	設計	メモリ
悪意ある攻撃	安全性 (safety)	実装・単体試験	ファイルシステム
ハードウエア故障	整合性 (integrity)	結合試験	通信
	保全性 (maintainability)	流通	入出力
		運用	電力
		保守・更新	システム全体
		廃棄・再利用	

図5: 定性評価項目

上図に、定性評価項目を示す。これは、ディペンダビリティ支援機構やツールの効果を、時間、空間、対象など複数の軸で評価する物である。オペレーティングシステムや機器の開発者は、ディペンダビリティ支援機構やツールそれぞれを定性評価項目と照合し、満たすと思われる性質をチェックしていく。次に開発者は、定性評価結果に対してベンチマーク、検証、フォルトインジェクション等の結果により根拠を与える。このとき、Goal Structuring Notation(GSN)をディペンダビリティ評価に特化させた、Dependability Case (D-Case)を用いる。下図に、D-Caseの例を示す。

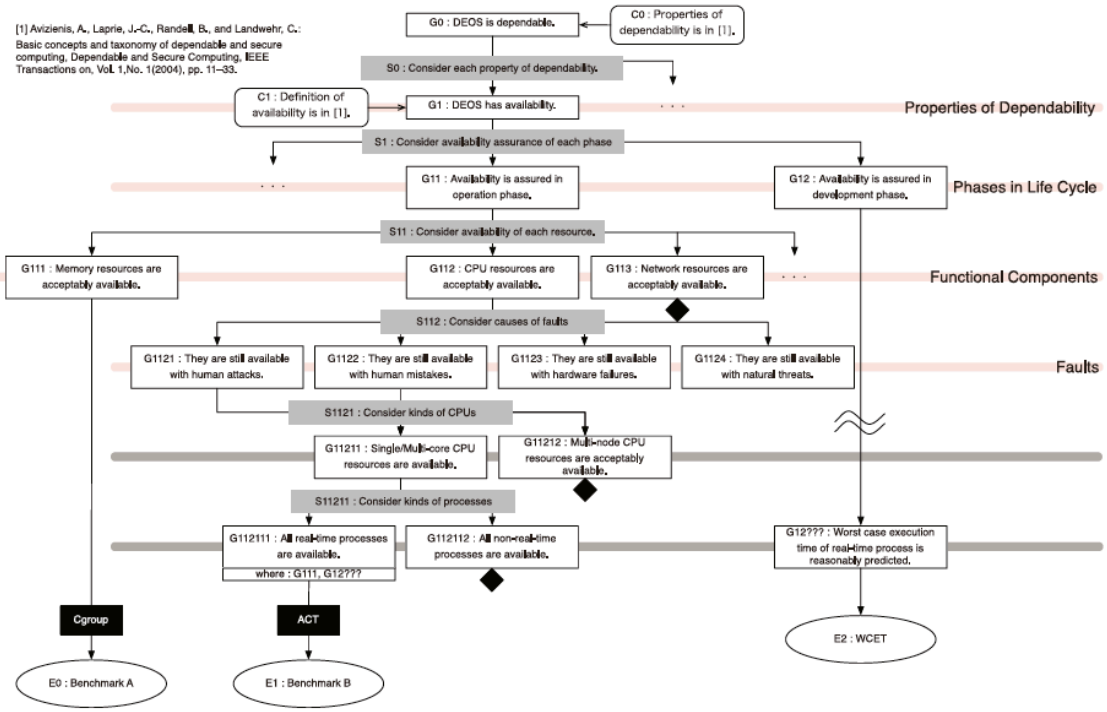


図6: D-Caseの例

本メトリクスでは、ディペンダビリティ支援機構やツールなどの要素技術とゴールとの関係を明確化できるよう、既存のGSNを拡張した。またゴールの分割方法と配置方法に制約を設け、ディペンダビリティに関する議論をより網羅的に行えるようにした。

今後、本メトリクスに関して以下の拡張を検討している。

・本メトリクスをシステム動作中のリアルタイムなディペンダビリティ評価に利用できるよう、拡張する必要がある。システム利用者が指定したディペンダビリティに関するポリシをモニタリングして、その結果に基づいてシステムの挙動を変更するフィードバックループの実現を検討している。

・本メトリクスでは、ディペンダビリティ阻害要因が発現した際のリスク分析を行えない。ディペンダビリティ支援機構同士の重要度を比較できるよう、実システムにおける阻害要因の影響度合いを考慮できる仕組みを検討している。

・本メトリクスでは D-Case を定量化する方式を決定していない。そのため、ディペンダビリティ支援の量的な評価を行えない。今後、各ディペンダビリティ支援機構がもたらす効果とそのオーバーヘッド等の副次効果から、オペレーティングシステム全体のディペンダビリティ支援の量を算出する方式を検討する。