戦略的創造研究推進事業 CREST
研究領域
「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」

研究課題「マイクロユビキタスノード用
ディペンダブル OS」

研究終了報告書

研究期間　平成 18 年 4 月 1 日～平成 24 年 3 月 31 日

研究代表者：徳田 英幸
(慶應義塾大学 環境情報学部 教授)

§1　研究の実施概要

(1)　実施概要

情報家電機器等で利用されているミッドレンジのハードウェアプラットフォームだけではなく、さらに小型軽量で、エネルギー効率の良い電池駆動可能なマイクロレンジのハードウェアプラットフォーム (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。本研究は、研究開始当初は (1) 複数種類のネットワークインタフェースを持つ次世代オープン端末のためのディペンダブル通信機能（研究課題 1)、ならびに (2) バッテリ電力の詳細なロギングと予約に基づくディペンダブル消費電力管理機能（研究課題 2）を研究課題として定めていたが、領域全体の方針転換を踏まえ、(3)D-Case およびディペンダビリティメトリクスを追加的に研究課題に定めて実施した。

ディペンダブル通信機構は、以下の 2 つの観点において組み込み機器の運用フェーズのディペンダビリティ支援に貢献する。まず、ネットワーク通信を行う組み込み機器が急速に増えている中で、本機構はトランスポート層での通信路の冗長化、および移動時の切断時間短縮により、ネットワークのアベイラビリティを向上させる。次に、アプリケーションごとに送信データの重要度を指定可能とすることにより、トランスポート層での無駄なオーバーヘッドを回避する。本機構では、トランスポート層プロトコル SCTP の改善を主なアプローチとしている。平成 20 年度までに、単一の無線ネットワークインタフェースを持つ組み込み機器が、互いに重なる無線 LAN 通信区間を横切るときに発生する通信切断時間を最小化する技術を開発し、その成果をまず Internet Draft として仕様を公開し、平成 20 年度までに FreeBSD オペレーティングシステム上に実装した。平成 23 年度までに Linux ソースツリー内での実装を完了した。それらの実装は **Linux、FreeBSD オペレーティングシステムに組み込まれ、実用化を完了**した。

ディペンダブル消費電力管理機構は、以下の 2 つの観点において組み込み機器のディペンダビリティ支援に貢献する。まず本機構は、消費可能な有限の電力において、組み込み機器の所定時間の動作 (アベイラビリティ) を保証する。電池駆動機器においては、全電池容量に対してアベイラビリティ保証の時間を指定できる。AC 駆動機器においては、所定の電力消費量に対して同様のことを行える。すなわち、電力消費の最適化が可能となる。また本機構は、組み込み機器の開発フェーズ、運用フェーズ、および保守・更新フェーズのディペンダビリティ支援を行うことを目的とする。開発フェーズにおいては、開発対象機器上で動作しうるプロセスのリストが判明しているとき、それらのプロセスを一定時間、一定負荷未満で動作させられる電力量を見積もるツールを検討する。運用フェーズにおいては、電源電力の監視とプロセスの選択的停止により、アベイラビリティ支援を行う。保守・更新フェーズにおいては、開発時に見積もった挙動と、実際の挙動との差異をログから抽出するツールを検討する。同ツールを用いることにより、電源の劣化を検出し、ハードウエアフォルトを早期に発見可能となることが期待できる。これまで、動作中プロセスが各デバイス（CPU やネットワークインタフェース等）を使用した時間や回数をカウントし、その比率に応じて消費電力量を分割する方式で、プロセスごとの消費電力計測を実現した。現在のプロトタイプは、XScale プロセッサを搭載した Linux ボードに Texas Instruments 製電力供給 IC を接続したハードウエアに依存して動作している。平成 21 年 8 月までに、X86 プロセッサを搭載した PC 上で、ACPI を用いて消費電力を計測し、それをプロセスごとの消費電力に分割する方式を実現する。

D-Case およびディペンダビリティメトリクスは、以下の 2 つの観点において組み込み機器のディペンダビリティ支援に貢献する。まず D-Case は、組み込み機器あるいはそれを含むシステムのディペンダビリティを論理的に示し、その証拠を与える仕組みである。これを用いてシステム発注者と開発者、運用者等のステークホルダが相互に議論し合意形成を行える。次にディペンダビリティメトリクスは、D-Case の記述内容に基づいて組み込み機器あるいはそれを含むシステムのディペンダビリティ達成状況を定量的に表す。これを用いてステークホルダはディペンダビリティに関する議論を客観的に行える。

(2) 顕著な成果

1. ディペンダブル通信機構 (実用化)

概要： トランスポートプロトコル SCTP を拡張して切断耐性の高い通信機構を実現した。FreeBSD7.0 Distribution および Linux 2.6 に組み込まれ、全世界で配布されている。

2. ディペンダブル消費電力管理機構 (試作品)

概要： バッテリ駆動 Linux 機器の消費電力を細粒度に管理、制御する機構を実現した。pSurvive システムとして Android MarketPlace で公開している。

3. D-Case およびそれに付随するツール群 (試作品))

概要： ディペンダビリティに関する合意形成ツールとして D-Case を策定し、そのエディタをはじめとするツールを他チームと協力して構築した。東京大学石川チームがホストするウェブサーバにおいて公開している。

§2　研究の進捗状況

(1)　当初の研究構想

本研究では、ネットワーク接続可能な情報家電機器や無線センサノード、携帯電話や PDA 等の携帯型計算機等をマイクロレンジ組込み機器と呼び、それらで共通に動作可能な、実時間性、高信頼性、高安全性、高可用性を持つオペレーティングシステムの基盤を、オープンソース OS である Linux を拡張して実現することを当初の目標とした。同基盤を提供することで、それらの機器を相互接続したアプリケーション開発の容易化と、それらのアプリケーションにおける実時間性や高信頼性、高安全性、高可用性といったディペンダビリティ向上をねらいとした。　具体的には、LinuxOS をベースとし、計算資源の限られた小型計算機上で動作するアプリケーションに対して、実時間制御機構、電力管理機構、耐フラジャイル通信支援機構を提供するための拡張を行い、より予測・解析可能な LinuxOS を実現する。特に、乾電池数本の大きさで実装された組込みノード（乾電池ノード）を主要ターゲットとし、その上で上記機能を実装し、さまざまなアプリケーションに対して実証実験を行う。これにより、上記のようなハードウエアプラットフォームを、乾電池数本で、時間と場所を問わず多用途に実用できるようにする。このような観点で、本研究で実現する拡張 Linux オペレーティングシステムをディペンダブル uLinux (d-uLinux) と呼ぶ。また、d-uLinux を搭載し、乾電池で実行可能なハードウエアプラットフォームを「乾電池 d-uLinux ノード」と呼ぶ。

研究開始後 3 年後をめどに Linux カーネル、電力管理・ネットワーク帯域管理・CPU 管理を含む統合資源管理ローダブルカーネルモジュールを含み、インテル IA32 アーキテクチャ、ARM アーキテクチャで動作するプロトタイプの構築を、また研究終了時に Linux カーネル、電力管理・ネットワーク帯域管理・CPU 管理を含む統合資源管理モジュールを含み、インテル IA32 アーキテクチャ、AMD64 アーキテクチャ、ARM アーキテクチャで動作するシステムの構築を計画していた。

(2)　新たに追加・修正など変更した研究構想

本領域では、チーム間の連携や研究領域全体でのゴール再設定などに基づき、当初計画の修正および追加を行った。

①　ターゲットの変更

研究領域全体として、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開発ならびに実用化から、オープンシステム用ディペンダビリティ開発・運用プロセス・アーキテクチャの策定に、ゴールが変更された。これに伴い、本研究が当初構想していたマイクロレンジ組込み機器をターゲットとした研究開発から、より一般的なシステムにおいて実用可能な機構の研究開発へとターゲットを変更した。具体的には、上述した無線センサノードや乾電池で実行可能なハードウエアプラットフォームを研究のフォーカスから外し、バッテリ駆動型携帯機器や携帯型計算機、自動改札機やチケット発券機などの非携帯型組込み機器を含むより一般的な組込み機器をターゲットとした。

②　研究課題の追加

オープンシステム用ディペンダビリティ開発・運用プロセス・アーキテクチャの策定に領域のフォーカスが移されたことに伴い、当初構想していた要素技術の開発に加え、D-Case およびディペンダビリティメトリクスの研究開発を課題に加えた。本研究チームは、D-Case サブコアチームの立ち上げと D-Case の本質的な考え方の確立、それを応用したディペンダビリティ定量評価手法の提案を行うこととした。

§3　研究実施体制

(1)　「徳田」グループ（慶應義塾大学 SFC 研究所）

①　研究参加者

氏名	所属	役職	参加時期
徳田　英幸	慶應義塾大学 環境情報学部	教授	H19.4
高汐　一紀	慶應義塾大学 環境情報学部	准教授	H19.4 H24.3
中澤　仁	慶應義塾大学 環境情報学部	講師	H19.4 H24.3
岩井　将行	慶應義塾大学大学院 政策・メディア研究科	講師	H19.4 H20.3
由良　淳一	慶應義塾大学大学院 政策・メディア研究科	助教	H19.4 H22.8
青木　崇行	慶應義塾大学大学院 政策・メディア研究科	博士課程 4 年	H19.4 H20.9
米澤　拓郎	慶應義塾大学大学院 政策・メディア研究科	助教	H23.4 H24.3
榊原　寛	慶應義塾大学大学院 政策・メディア研究科	博士課程 3 年	H19.4 H20.3
米澤　拓郎	慶應義塾大学大学院 政策・メディア研究科	博士課程 3 年	H19.4 H20.3
堀川哲郎	慶應義塾大学大学院 政策・メディア研究科	修士課程 1 年	H23.4 H24.3
間　博人	慶應義塾大学大学院 政策・メディア研究科	博士課程 6 年	H19.4 H21.3
徳田　英隼	慶應義塾大学大学院 政策・メディア研究科	修士課程 2 年	H19.6 H20.3
本多倫夫	慶應義塾大学大学院 政策・メディア研究科	博士課程 2 年	H20.4 H24.3
森 雅智	慶應義塾大学大学院 政策・メディア研究科	博士課程 2 年	H20.4 H24.3
斉藤 匡人	慶應義塾大学大学院 政策・メディア研究科	博士課程 6 年	H21.4 H21.12
野沢 高弘	慶應義塾大学大学院 政策・メディア研究科	修士課程 2 年	H21.4 H23.3
米川 賢治	慶應義塾大学大学院 政策・メディア研究科	修士課程 1 年	H22.4 H24.3
生天目直哉	慶應義塾大学大学院 政策・メディア研究科	博士課程 2 年	H23.4 H24.3
井村和博	慶應義塾大学大学院 政策・メディア研究科	修士課程 1 年	H23.4 H24.3
加藤碧	慶應義塾大学 環境情報学部	4 年	H23.4 H24.3
伊藤 友隆	慶應義塾大学大学院 政策・メディア研究科	助教	H23.4 H24.3

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、D-Case 策定、ディペンダビリティメトリクス (コアチームにおける作業)

§4　研究実施内容および成果

(1)　慶應義塾大学　徳田グループ

(1)-1 ディペンダブル通信機構

①　研究実施内容及び成果

車や携帯マルチメディア端末、携帯ゲーム機など、多くのモバイルノードが無線ネットワークを介してインターネットに接続されるようになった。将来のインターネット技術においては多くの利用者が移動しながらインターネットに接続することが予想されるため、移動支援技術が必要となる。異なる IP アドレス空間を持つ別々のキャリアが提供する無線アクセスポイント間を移動する場合、モバイルノードではその都度無線インタフェースに新しい IP アドレスが設定される。しかし IP アドレスの変更は、ネットワーク切り替えのオーバーヘッドにより、コネクティビティ切断を引き起こし、ネットワークのアベイラビリティが低下する。本研究では特に次のような２つのシナリオについて考える。

1. **無線エリア外を通過した場合**
   モバイルノードが他の無線ネットワークへ移動する際に、無線が届かないエリアを通過した場合 (図 1) である。つまり、ネットワーク A とネットワーク B の範囲が重なっていないため、モバイルノードはネットワーク A を離れてからネットワーク B に到着後、IP アドレスが付与されるまで他のノードとの通信は不可能である。

2. **同無線インタフェース上でネットワークを切り替えた場合**
   ネットワーク A と B の範囲が重なっている時 (図 2)、ネットワーク A に接続されているモバイルノードはネットワーク A を出ずにネットワーク B にも接続可能である。しかしモバイルノードは、接続先の無線ベースステーションを切り替える際のリンクレイヤのハンドオフ時や、上述した方法により IP アドレスが新規に設定されるまでネットワークから最大数秒切断される。

本研究では、IP アドレスの変更を要因とする送信遅延の短縮を目的とした、新しい SCTP 再送手法を提案する。SCTP は、広く使われている接続指向のトランスポートプロトコルである TCP の提供する機能に加え、高度な機能を提供する新しいトランスポートプロトコルである。SCTP は IETF によって標準化されており、FreeBSD や Linux、AIX、Solaris などの主要な OS 上で実装されている。このように、将来のディペンダブルなモバイル通信のためのインターネット技術の基礎技術として SCTP は有望視できる。本手法は、1. 双方向性、2. 自己完結性という特徴を提供する。前者は、送信遅延の短縮が、移動ノードから通信相手に対してのデータ送信だけでなく、通信相手から移動ノードに対してのデータ送信にも有効であることを意味する。後者の特徴は、本手法が SCTP 上のイベントのみ利用するため、SCTP 以外のレイヤを変更する必要が無いことを意味する。

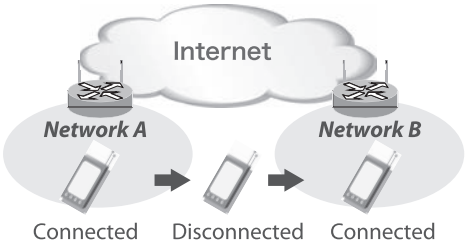

図 1: 無線エリア外を通過した場合

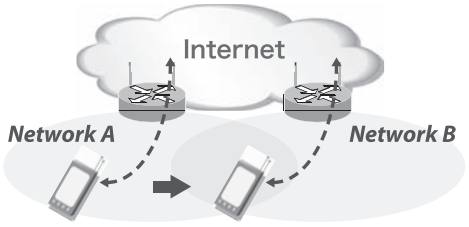

図 2: 無線ネットワークを切り替えた場合

1　問題分析

無線ネットワーク間をモバイルノードが移動する際に起こる SCTP の送信遅延について述べる。

1.1　アドレス再設定操作

SCTP (Stream Control Transmission Protocol)[1] は TCP に比べ新しいトランスポートプロトコルである。SCTP はコネクション指向で信頼性のあるデータ転送を提供する。2 つの SCTP エンドポイント間の接続は**アソシエーション**と呼ばれ、両端の IP アドレスおよびポート番号の組み合わせによって識別される。エンドツーエンド通信では、両エンドは IP アドレスによって識別される。IP アドレスが変更されると、通常はコネクティビティが切断される。これは、ノマディックコンピューティングにおいてよく知られた問題である。SCTP における ADDIP 拡張はこの問題を解決する。図 3 の *Address Reconfiguration Operation in SCTP* に、モバイルノードが別のネットワークに移動した場合での、SCTP アソシエーションのアドレス再設定の流れを示す。

図 3 では、*IP_ADDR_A* を持つモバイルノード (MN) が *IP_ADDR_C* を持つ通信相手 (CN) と無線ネットワークの SCTP アソシエーションを介して通信している。その後 MN が移動すると、MN 側の SCTP エンドポイントは現在の IP アドレス (*IP_ADDR_A*) が無効になったことを検知する。これは、無線範囲から出た場合 (シナリオ 1) か、別の無線ネットワークに接続された場合 (シナリオ 2) に起こる。SCTP の仕様では、エンドポイントへ最後に残ったアドレスはアソシエーションから削除されないため、この段階では MN 側のエンドポイントは SCTP アソシエーションから *IP_ADDR_A* を削除できない。IP アドレスのインタフェースからの削除検知は実装依存であり、リンク状態や新しい IP アドレスの設定によって行われる。

その後、MN が新しい別のネットワークの IP アドレス (*IP_ADDR_B*) を取得し、MN 側 SCTP エンドポイントがそれを検知した場合、エンドポイントは ASCONF を送信する。ASCONF には、追加 IP アドレス、削除 IP アドレス、および新たなプライマリ宛先の 3 つのパラメータが含まれる。削除 IP アドレスパラメタには、削除された IP アドレス (*IP_ADDR_A*) が、追加 IP アドレスパラメータには、追加された IP アドレス (*IP_ADDR_B*) が含まれる。

CN が ASCONF を受信すると、MN の新たなアドレスへの到達性を確認するため、ASCONF-ACK に同梱した HEARTBEAT よって確認応答を行う。MN は ASCONF-ACK を受信することで、新たな IP アドレスからのデータ送信が可能となったことを確認できる。その後 HERTBEAT-ACK の送信によって CN から送信された HERTBEAT への確認応答が行われ、CN は MN の新たなアドレスの到達性を確認でき、そのアドレスへのデータ送信が可能となる。

1.2　再送の挙動における問題

図 3 において、上部に示される灰色部分はネットワークの切断期間である。切断期間は、図 3 の下部に灰色で示されるように、SCTP アソシエーションにおける送信遅延を引き起こす。この現象は、輻輳制御機構における再送タイムアウトに関する挙動が原因である。

切断期間中は、MN 側 SCTP エンドポイントはデータ送信に失敗する。第 1.1 節で述べたアドレスの再設定操作後、エンドポイントは確認応答されていないデータを再送する必要があるが、通信相手側への到達性が復旧した場合にも、これらのデータはすぐには再送されない。次の再送を行うには、MN 側エンドポイントにおいて次の再送タイムアウト (RTO) が起こるまで待機する必要がある。RTO の値はタイムアウト毎に 2 倍に設定される。そのため、切断期間中に複数回再送タイムアウトが生じた場合、次の再送タイムアウトまでの時間が非常に長くなる可能性がある。以上が、接続復旧後に生じる送信遅延を引き起こす原因である。

同様に、CN 側 SCTP エンドポイントも MN 側の切断期間中はデータ送信に失敗する。そのため、CN 側 SCTP エンドポイントは、ASCONF により MN の新規アドレスを通知され、HEARTBEAT により到達性が確認された後、新規アドレスに対して確認応答されていないデータを送信する必要がある。再

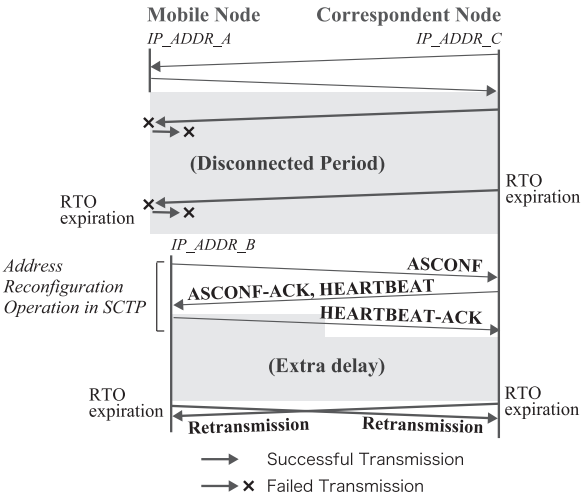

図 3: 切断期間を含む移動中の SCTP の動作

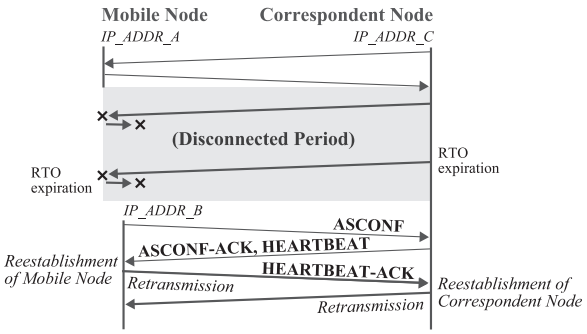

図 4: 理想的な再送時の挙動

送は次の再送タイムアウトの際に行われるため、CN 側エンドポイントにおいても MN 側エンドポイントと同様に、切断期間中に起こる複数回の再送タイムアウトが非常に長い待機時間を生じさせうる。

RTO の値はラウンドトリップタイム (RTT) を元に算出されるが、1 秒未満の場合、TCP の再送タイマ [2] の推奨時間である 1 秒に設定される。また RTO の最大値は、60 秒より大きく設定されることが推奨されている。SCTP BSD カーネルの実装 [3] はこの推奨に従っており、RTO の最小値を 1 秒に、最大値を 60 秒に設定している。以下の説明では、議論を簡略化するため、これらの値を用いる。

上述の結果、再送成功までの時間は以下のように表せる。

$$RTO_0 \times 20 + RTO_0 \times 21 + ... + RTO_0 \times 2^{N-1} = \sum_{n=0}^{N-1} RTO_0 \times 2^n \quad (1)$$

RTO_0 は、最初の再送タイムアウトまでの RTO の値を表す。また N は、再送成功までに起こる再送タイムアウトの数を示す。N は切断期間と RTO_0 によって決定され、以下の式で表される。

$$\sum_{n=0}^{N-1} RTO_0 \times 2^n = RTO_0(2^N - 1)$$

$$RTO_0(2^{N-1} - 1) \leq DisconnectedPeriod < RTO_0(2^N - 1)$$

$$N = \left\lfloor log_2 \frac{DisconnectedPeriod + RTO_0}{RTO_0} \right\rfloor + 1 \quad (2)$$

切断期間後の遅延は、次の再送タイムアウトまでの時間であり、以下の式によって求められる。

$$ExtraDelay = \sum_{n=0}^{N-1} RTO_0 \times 2^n - DisconnectedPeriod \quad (3)$$

以上より、切断期間後の再送タイムアウトが長くなるに従い、余分な遅延が長くなりうる。最悪の場合、余分な遅延は RTO の値の長さ ($RTO_0 \times 2^{N-1}$) とほぼ同値になる。

2 新たな再送手法

本研究では、2 つの再送トリガを用いる。本手法ではコネクティビティが失われた際にロスしたパケットを MN のコネクティビティが復旧すると同時にデータ送信側が再送を行う。図 4 に本研究が提案する理想的な再送時の挙動を示す。

2 つの再送トリガは両エンドホストに独立して実装される。本手法の 1 つ目のメリットとして両エンドノードは互いに依存しておらず、独立して動作することである。したがって通信に用いる両方のホストによって本手法がサポートされる必要はなく。データ送信側が本手法をサポートしていれば余分な遅延を減らすことが可能である。2 つ目のメリットは SCTP の機能を利用してアドレスの再設定を行っていることである。コネクティビティの復旧は、MN において ASCONF-ACK を受信し、CN において HEARTBEAT-ACK を受け取ることにより確認される。つまり、他のレイヤへの変更は一切なく、プロトコルの仕様に準拠している。

3 実装

本研究において提案した再送手法は、FreeBSD や Mac OS で実装されている SCTP BSD kernel implementation において実装を行った。この実装は SCTP の基本機能やいくつかの拡張を提供する。例えば、ADD IP 拡張、SCTP Authentication、Partial Reliability 拡張、Advanced Socket API などである。mobile node と correspondent node の機能の両方とも、新しい sysctl パラメータである net.inet.sctp.mobility_fasthandoff により使用可能である。SCTP BSD kernel implementation は定期的に FreeBSD の CVS レポジトリにマージされるため、我々の実装も FreeBSD 7.0 以降で使用可能である。すなわち、**FreeBSD オペレーティングシステム上では実用化済み**である。

一方、Linux では SCTP はカーネルモジュールとして実装されている。本研究の提案手法は、SCTP スタックがアドレスの変更の際自動的に ASCONF を送信する機能である AUTO_ASCONF を用いて実装されるが、Linux の SCTP 実装にはこの AUTO_ASCONF の機能が実装されていないため、まずそこから着手している。この機能は、現在カーネルのコンパイルオプション (CONFIG_SCTP_AUTO_ASCONF) および sysctl パラメータ (net.sctp.auto_asconf_enable) として実装している。この実装を平成２１年度中に終了させ、研究期間終了までに Linux ソースツリーにマージし、実用化することを検討している。

4 評価

本章では、前述したシナリオに基づく実験評価により、本研究課題の成果を示す。今回の評価はオリジナルの SCTP (SCTP BSD kernelimplementation) と提案したアルゴリズムで、移動ノードと対応ノード間における SCTP アソシエーションの転送速度の伸びによって行った。

4.1 評価環境

評価環境マシンとして Mobile Node (MN) には 1.3GHz の Pentium M プロセッサと 1280MB のメモリ、無線 LAN インタフェースを 1 つ搭載した IBM Thinkpad X40 を用いた。また、Correspondent Node (CN) には 1.0GHz の Pentiun M プロセッサと 512MB のメモリを搭載した IBM Thinkpad X40 を用いた。 MN 、CN 共に使用 OS は FreeBSD 6.1 である。

評価用ネットワークには 3 つの IPv6 ネットワークを使用した。Network 1 と 2 はそれぞれ IEEE801.11b の無線 LAN アクセスポイントであり、おおよそ 30 cm 程離して設置した。そのため、それぞれの無線エリアは重複している。 MN は最初 Network 1 へ接続しているが、通信の途中で Network 2 へ接続の切り替えを行う。 Network 3 は有線 LAN で CN が接続されている。中継ノードは Dummynet が設置してあり、MN - CN 間の RTT を 20ms と設定した。

本実験では MN から CN へ 8MB のデータ転送を行い、またその逆の転送も行った。送信側ノードは断続的に 1408Byte のデータパケットをソケットに対し、システムコールの write() を用いて送信を行った。受信側ノードはソケットに対してシステムコールの read() を用いて受信した。MN からのデータ転送と、CN からのデータ転送は同時ではなくそれぞれ単独で転送実験を行った。

前述したシナリオは以下のようにエミュレーションする。はじめに、MN は Network 1 に接続され、通信開始から 5 秒後に Network 1 とのコネクティビティを切断する。T 秒後、MN を Network 2 へ接続して新たな IPv6 アドレスを取得する。これにより図 1 で示すような、移動端末が無線 LAN の電波の届かない場所を通り、リンクダウンを起こした状態をエミュレートしている。一方で、図 2 に示すように、T の値を 0 にすることでリンクのダウン時間を発生させずに無線ネットワークを切り替えることもエミュレートしている。

これまでの実験から計測されている、無線ネットワークの切り替えにかかる先天的なデバイスのオーバーヘッドが約 0.7 秒、オリジナルの SCTP 実装における最小 RTO の値は 1 秒である。リンクのダウン時間を、0, 3, 6, 9 秒と変化させながら実験を行った。効果が出にくいと予想されるのは T を 0 秒または 6 秒と設定した時であり、逆に効果が出やすいと予想されるのは T を 3 秒、もしくは 9 秒と設定した時である。評価項目は以下に示す 4 つの値であり、それらを測定することにより評価を行う。

1. **再送までにかかる遅延時間**
   受信側ノードにおいて MN のマイグレーションを行っている際の受信データの途切れていた時間を測定する。

2. **コネクティビティの断続時間**
   この値は、設定したリンクのダウン時間、MN 側での新しい IP アドレスの設定時間を含んでいる。

3. **転送時の余分な遅延**
   転送時の余分な遅延は、転送時に生じた遅延の合計からコネクティビティの断絶時間を引いたものである。本提案手法の効果は、この値に顕著に表れる。

4. **転送に要した合計時間**
   これは、受信側ノードにおいて 8MB のデータの受信に要した合計時間である。この値は転送に生じた遅延の程度を比較することに適している。

表 1: MN からの送信結果

リンクダウン	0 s		3 s		6 s		9 s	
	本手法	SCTP	**本手法**	SCTP	**本手法**	SCTP	**本手法**	SCTP
切断時間	0.644 s	0.642 s	3.731 s	3.699 s	6.576 s	6.628 s	9.654 s	9.695 s
送信遅延	**0.691 s**	1.071 s	**3.795 s**	7.061 s	**6.637 s**	7.068 s	**9.737 s**	15.062 s
余分な遅延	**0.047 s**	0.429 s	**0.064 s**	3.362 s	**0.061 s**	0.440 s	**0.083 s**	5.367 s
合計受信時間	**13.387s**	13.888 s	**16.484 s**	19.561 s	**19.455 s**	19.523 s	**22.558 s**	28.438 s

4.2 実験結果

表 1,2 は実験結果を示している。数値は全て 5 回ずつ実験を行って取得した数値の平均値となっている。コネクティビティの切断時間を 0, 3, 6, 9 秒と変更させて実験を行ったが、表中では” リンクダウン” の列に切断時間を示した。提案したアルゴリズムを用いた SCTP とオリジナルの SCTP を比較して、リンクのダウン時間が 3 秒、9 秒であった際、合計受信時間はそれぞれ 3 秒、6 秒も短縮された。一方で、リンクのダウン時間が 0 秒、6 秒であった際、転送に要した合計受信時間は大きな短縮とはならなかった。以下に実験結果の詳細を述べる。

表 2: CN からの送信結果

リンクダウン	0 s		3 s		6 s		9 s	
	本手法	SCTP	**本手法**	SCTP	**本手法**	SCTP	**本手法**	SCTP
切断時間	0.599 s	0.672 s	3.681 s	3.621 s	6.531 s	6.596 s	9.738 s	9.645 s
送信遅延	**0.643 s**	1.001 s	**3.725 s**	7.001 s	**6.576 s**	7.001 s	**9.785 s**	15.018 s
余分な遅延	**0.044 s**	0.329 s	**0.044 s**	3.380 s	**0.045 s**	0.405 s	**0.047 s**	5.373 s
合計受信時間	**14.092 s**	14.361 s	**17.214 s**	20.164 s	**20.059 s**	20.209 s	**23.610 s**	29.983 s

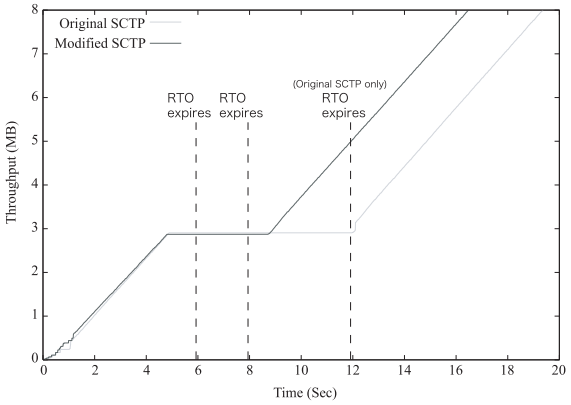

図 5: 3 秒間のリンクダウンにおける CN からのデータ転送

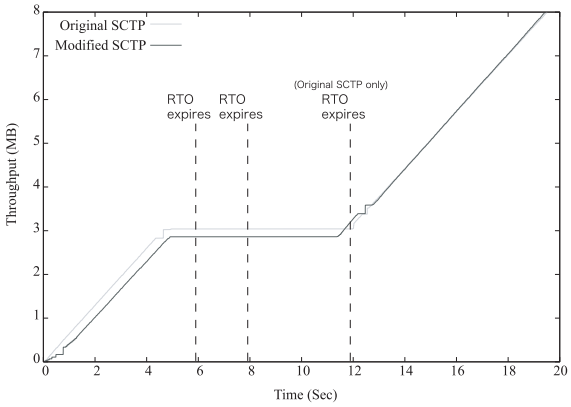

図 6: 6 秒間のリンクダウンにおける CN からのデータ転送

最小 RTO 値が 1 秒かつ、 RTT が 20ms であった時、再送タイムアウトはコネクティビティが断絶されてから 1, 3, 7, 15 秒後に生じる。リンクのダウン時間が 3 秒、9 秒であった場合、コネクティビティ復旧後最初に発生する再送タイムアウトはそれぞれ 7 秒後、15 秒後となる。本手法を用いた場合、再送タイムアウトを待たずに再送を開始するため大幅に遅延時間を減らすことができる。

図 5 は MN のコネクティビティの切断時間が 3 秒であった場合の CN からデータを送信した際のスループットを示している。この図は 5 回の実験の平均の数値ではなく、5 回の中の 1 つである。この実験を行った際、コネクティビティ断絶時間はオリジナル SCTP を用いた場合は 3.774 秒で、提案アルゴリズムを用いた場合は 3.851 秒だった。オリジナル SCTP は、コネクティビティの断絶時間が 3.774 秒であったにもかかわらず、再送までの遅延が 7.075 秒もかかっている。これはコネクティビティが失われている間に再送タイムアウトを起こしてしまっているからである。最小 RTO 値が 1 秒で、コネクティビティを失っている間に 2 回再送タイムアウトが発生していたとき、コネクティビティが復旧しても RTO 値は 4 秒も残っている。7.075 秒の遅延は、 3 回の再送タイムアウトの蓄積によるもので、具体的には、1 秒、2 秒、4 秒の合計の値である。一方で、提案したアルゴリズムを加えた SCTP での転送では、ASCONF と HEARTBEAT-ACK を受け取るとすぐに再送を開始する。そのため再送までの遅延が 3.912 秒であり、コネクティビティの断絶時間とかなり近い値となっている。結果として合計受信時間は、オリジナル SCTP が 19.370 秒かかったのに対し、提案手法を用いることで 16.476 秒へ大幅に短縮することができた。

一方、0 および 6 秒間のリンクダウン時間ではあまりパフォーマンス向上がみられない。これは切断期間が終了後の最初の再送タイムアウトがすぐに (1 秒) 来るためで、比較的小さなパフォーマンス向上になる。図 6 に、6 秒間のリンクダウンをまたぐネットワーク切り替えにおける CN から MN へのスループットを示す。本実験において実際に切断された時間は、通常の SCTP では 6.626 秒であり、本手法を適用した場合には 6.417 秒であった。送信遅延は、元の SCTP では 7.028 秒。本手法では 6.460 秒であ

る。元の SCTP では、6.626 秒の切断時間後、再送タイムアウトが切断開始後 7 秒で起きた。これは、ASCONF および HERTBEAT-ACK の受信後すぐに確認応答されていないデータが順次再送されていることを示す。したがって、送信遅延の差は上述の場合に比べ小さく、総受信時間は通常の SCTP の場合は 19.512 秒、本手法の場合は 19.429 秒となった。

ディペンダブル通信機構は、領域全体において次のように位置づけられる。本機構は、以下の 2 つの観点において組み込み機器の**運用フェーズ**のディペンダビリティ支援に貢献する。まず、ネットワーク通信を行う組み込み機器が急速に増えている中で、本機構はトランスポート層での通信路の冗長化、および移動時の切断時間短縮により、ネットワークの**アベイラビリティ**を向上させる。これは、ネットワークの **failure-proof** である。次に、アプリケーションごとに送信データの重要度を指定可能とすることにより、トランスポート層での無駄なオーバーヘッドを回避する。これは、ネットワークの **performance-degradation-proof** である。

次に、本研究の新規性ならびに画期的な点は以下の通りである。ここでは関連研究との比較において論じる。移動端末におけるネットワークの availability 支援には次のような関連研究がある。まずトランスポート層におけるアドレス変更サポートに関して、Migrate TCP[4] と TCP-R[5] は、実行中の TCP コネクションにおける動的な IP アドレスの変更を実現する。mSCTP[6] は ADDIP 拡張を用いた SCTP のモビリティサポートを提案 している。しかしこれらの目的はアドレス変更時における TCP コネクションのあるいは SCTP セッションの維持のみ であり、パフォーマンスについては十分に考慮していない。これに対して本研究では再接続後の遅延を最小化して、**移動端末におけるネットワークのアベイラビリティを大幅に向上**させた。

次にネットワーク層におけるアドレス変更サポートに関して、HIP[7]、Mobile IPv6[8]、VIP[9] は実際にインタフェースに設定される IP アドレスとは別の識別子をトランスポート層に提供し、トランスポート層にその識別子を用いて通信させることで、IP アドレスの 変化をトランスポート層に対して隠蔽する。そのため、IP アドレスが変化した際も TCP コネクションは継続することができるが、移動前に使用していた IP アドレスの無効化によるコネクティビティの切断をも隠蔽されてしまい、コネクティビティ復旧後に大きな遅延が発生しうる。従って本研究で実現した大幅なアベイラビリティ向上は期待できない。

TCP におけるパフォーマンスの改善に関して、Cáceres[10] らは Mobile IPv6 における IP アドレス変更通 知メッセージである Binding Update をトリガーにした高速再送を提案している。また、Schütz[11] らは、TCP が HIP 上で動作している状況 において、HIP レイヤにおける IP アドレス変更イベントをトリガーにした高速 再送を提案している。しかし、これらは TCP が Mobile IP あるいは HIP 上で動作していることを前 提にしているため、これらの有用性は TCP の性能だけでなく、その下の Mobile IP あるいは HIP の挙動に依存する。そのため、Mobile IP や HIP などのモビリティプロトコル毎の実装が必要になる。それに対して本研究では IP アドレスの変更を直接扱える SCTP にフォーカスし ているため、下位レイヤのモビリティプロトコル、およびその変更を必要とせず、**幅広い製品において応用可能**である利点がある。

最後に、SCTP におけるパフォーマンスの改善に関して、Chang[12] らは、Set Primary を含む ASCONF を受信した際に、 再送データではなく新たなデータを送信し、それに対する SACK からコネクティ ビティ切断中にロスしたシーケンスを検知し、高速再送する手法を提案している。しかし、この手法は再送データの他にまだ送信されていないデータを送信側が 持っている必要があり、そうでない場合には動作しない。また、この手法はモバイルノード側からのデータ送信に関しては考慮していない。これに対して本研究では、未送信のデータを必要とせず、**モバイルノード側、サーバ側両方からのデータ送信に対応**する。

② 研究成果の今後、期待される効果

本研究では、次世代トランスポートプロトコルである SCTP を用いた通信において、ASCONF パケットを用いて、1.) 端末のネットワーク間のモビリティをサポートする。また、2.) その際のハンドオーバ遅延を最小にする。1.) に関して、FreeBSD における実装は既に他者によって実装されていたため、2.) を FreeBSD に実装した。その実装は、FreeBSD 7.0 以降のカーネルに組み込まれている。

本プロジェクトでは、より広い実用化に向けて、1.) および 2.) を Linux 上にも実装した。Linux は FreeBSD と異なり、SCTP そのものの実装が遅れていたため、1.) の部分も実装する必要があった。1.) に関して、本プロジェクトでは複数のソケットタイプへの対応 (TCP style や UDP style)、複数のアドレスファミリへの対応、Sysctl による本機能の On/Off を実装した。

1.) および 2.) のプロトタイプ実装を完了し、2010 年より Linux カーネルのマージ作業を開始した。そのため、ネットワーク関連の新機能が取り込まれるカーネルソースコードのツリーである net-next-2.6 に対してまず 1.) のパッチを投稿した。その後、パッチの査読者より *setsockopt()* システムコールを通じての本機能の On/Off 機構の追加、複数のアドレス bind タイプへの対応、新たなデータ構造の定義、ユーザ空間からの割り込みに対する適切なロック、IPv6 関連の適切な関数の利用に関するコメントを得た。ここまでの実装は、net-next-2.6 ツリーだけでなく、mainline のカーネル (http://www.kernel.org) にも統合されている。

2011 年も引き続き何度かのパッチ投稿を行い、バグの修正、データ構造の最適化、関数への処理の切り分け、タイマー操作の最適化などを行った。また、2011 年度は、ハンドオーバ後の処理に関するバグ修正および 2.) に関する実装を行った。2.) の実装については、それが最初から含まれる形で 1.) を実装したが、いくつかの問題があった。まずは、ハンドオーバー後に、本来は固定ノード側から HEARBEAT パケットをやりとりし到達性の確認をする必要があるが (RFC5061)、固定ノードが、HEARTBEAT パケットをやりとりする前にデータ転送を開始してしまうバグを修正した。この修正は、8 月 25 日に net-next-2.6 ツリーにマージされた。また、2.) に関連して、ハンドオーバー後に、固定ノード側からの HEARTBEAT パケットが遅延し、データ転送再開までに時間がかかる問題があった。この問題の解決も 8 月 25 日に net-next-2.6 ツリーにマージされた。そのため、数ヶ月以内には mainline にマージされると思われる。残る課題として、ハンドオーバ後に、リンク切断の通知イベントの起こり方次第で、モバイルノードから固定ノードへのデータ転送においてハンドオーバが遅れる場合がある問題がある。この解決策は実装済みだが、現在 kernel.org 側のトラブルにより net-next-2.6 ツリーの同期ができないため、この問題が解決し次第パッチを投稿する。

(1)-2 ディペンダブル消費電力管理機構

① 研究実施内容及び成果

携帯電話やポータブルオーディオプレーヤ、センサノードといったマイクロユビキタスノードが急速に普及していく中で、問題となっているのがバッテリ管理である。バッテリ寿命はデバイスの性能を測る大きな指標である。バッテリ寿命が長ければそのデバイスの可用性は向上する。バッテリ寿命をデバイス単位ではなくアプリケーション単位で見た場合、マイクロユビキタスノードはより複雑な問題を抱えている。単機能デバイスの場合、バッテリの利用用途は一種類のため、容易にアプリケーションの駆動時間を予測することができる。しかし、スマートフォンを元とする、通信やセンシングなど多機能を有するマイクロユビキタスノードにおいては、こうしたアプリケーションを基にした駆動時間の予測は困難である。アプリケーションから見た場合、複数のアプリケーションが一つのバッテリを共有して使用するため、一つのアプリケーションだけに着目したバッテリ駆動時間を知るだけでは意味がないからである。携帯電話メーカーでは連続待受時間や連続通話時間といった形でバッテリ寿命の仕様を公開しているが、これではメールや Web ブラウザを使う場合にはどの程度バッテリが保つのかはわからない。

こうした背景から、マイクロユビキタスノードにおいては、利用者がバッテリ残量を見ながら自分でバッテリ管理をしているのが現状である。しかし、利用者自身が管理するのではちょっとしたミスが頻発し、携帯電話で TV を見ていたらバッテリが切れてしまい、重要なメールが見られなくなったり、電子マネーでの決済ができなくなったりしてしまうというリスクが常に付きまとう。こうしたアプリケーションの動作時間を保証できないという問題がマイクロユビキタスノードの可用性を下げ、信頼性を下げる要因となっている。

本研究では、アプリケーション毎の消費電力を細かく計測、予測し、そのデータを基にした指定アプリケーションの動作時間保証を実現するシステム **pSurvive** の開発を行う。

アプリケーション A、B、C（以下 A、B、C とする）が動作する環境において、A が動作時間保証を行いたいアプリケーションとし、B、C は利用者によって任意に起動されるものとする。こうした環境において、本研究では以下の形式で動作時間保証を実現する。

1. 将来起動される A の動作時間を N 分保証する
2. 現在起動中の A の残り動作時間を N 分保証する

二つの違いは動作保証を実行するタイミングの違いだけで、保証内容は同様である。利用者が A の動作時間を予約すると、pSurvive は A が N 分動作するための電力量を確保し、A 専用とする。A 専用に確保した電力量は、A 以外によって使われることはない。その後、B、C が起動してバッテリが A 専用に確保した分しか残らなくなると、pSurvive は B、C を停止する。それにより、A の動作時間を保証することができる。

本研究が達成されることによって、二つの成果が得られる。一つは現状のマイクロユビキタスノード環境に pSurvive を導入することにより、重要度の高いアプリケーションの可用性を高めることである。これにより利用者の利便性が向上する。もう一つはアプリケーションの動作時間保証によって、従来は実現できなかった新しいサービスが出現する可能性が生まれることである。特に、高可用性、高信頼性を要求されるサービスが実現可能となることが期待される。

1　機能要件

pSurvive は、**特定のアプリケーションの動作時間を予約することで、アプリケーションの動作時間保証を実現する**。これにより、予想外のバッテリ切れを回避し、利用者はアプリケーション単位での電力管理が可能となる。また、優先度逆転問題についても優先度の高いアプリケーションの動作時間を予約することで、優先度の低いアプリケーションの影響を回避することができる。

具体的な機能として、以下の機能が必要である。

- アプリケーションの動作時間を予約するための API
- 予約内容に基づいて保証対象のアプリケーションのための電力をキープする機構

一番目は例えば「将来起動するアプリケーション A を N 分間動作させるための電力を予約」や「現在起動しているアプリケーション B を残り M 分間動作させるための電力を予約」といった形式になる。具体的な予約方法については、利用者かアプリケーション自身が pSurvive に対して予約 API を呼び出す形式となる。

二番目は、pSurvive が予約要求された内容に合わせて電力をコントロールする部分となる。しかし、この機能を実現するためには、アプリケーションの動作保証のためにどれだけの電力量を残す必要があるのかを予め予測できる必要がある。現状のシステムでは各アプリケーション別の消費電力量が正確に分からないため、その前提として各アプリケーションの消費電力量を前もって知る必要がある。

これらの問題を解決するため、pSurvive はアプリケーション別消費電力計測・予測機構とアプリケーション動作時間保証機構の二つのコンポーネントで構成される。アプリケーション別消費電力計測・予測機構では各アプリケーションの消費電力量をモニタリングし、今までの電力消費状態を記録すると共に、将来単位時間あたりどのくらいの電力を消費するのかを予測する。アプリケーション動作時間保証機構では、消費電力予測の結果を基に、実際に保証対象アプリケーションのための電力をキープし、必要であれば他の保証対象外アプリケーションを停止させる。また、機能以外にシステムを普及させるための要件として、次の点も考慮する。

- ノードに追加のハードウェアを実装する必要が無く、ソフトウェアのみで実現されること
- 既存のアプリケーションに修正の必要がないこと

2　基本設計

前節で示した通り、pSurvive はアプリケーション別消費電力計測・予測機構とアプリケーション動作時間保証機構の二つで構成される（図 7）。

2.1　消費電力計測・予測機構

pSurvive では、アプリケーションの使用電力量を各アプリケーション、各デバイス別に計測し、それらの電力の和を全体の消費電力として扱う。まず、ノードが n 個のデバイスを持っていると考える。このデバイスとは、CPU やメモリ、ネットワークやセンサといった電力を消費するハードウェアを指し、各デバイスを $d_1, d_2, d_3, ..., d_n$ とする。そして、各デバイスの単位使用量あたりの消費電力を c [mW/units] とし、$c_1, c_2, c_3, ..., c_n$ とする。これは例えば液晶ディスプレイを 1 分間動作させるのに必要な電力といったものに対応する。この単位使用量の単位 unit とは、デバイスによって定義が異なる。例えば液晶ディスプレイといったデバイスでは使用時間が電力消費量の指標になるが、センサデバイスではセンシングする回数で使用電力が決まるため、時間ではなくセンシング回数となる。

この時、デバイスの消費電力の合計 E はデバイスの使用量を t [units] とすると、$E = ct$ [mW] で表される。そのため、各デバイスの使用時間を $t_1, t_2, t_3, ..., t_n$ とすると、ノード全体の消費電力量は式 4 で表される

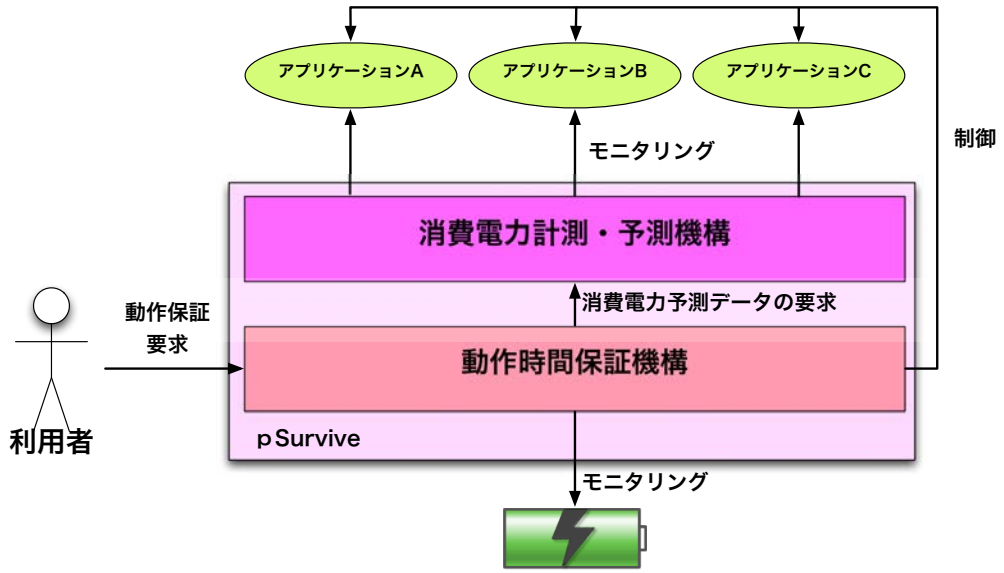

図 7: pSurvive システム構成図

$$E = \sum_{i=1}^{n} c_i t_i \tag{4}$$

次に、ノード上では m 個のアプリケーションが動作しているとし、$p_1, p_2, p_3, ..., p_m$ で表す。このアプリケーションはアプリケーションに対応している。アプリケーションは各デバイスを使用するため、アプリケーションの使った各デバイスの消費電力量の総和がそのアプリケーションの消費電力量であると考えられる。ここで、アプリケーション p がデバイス d を使用した使用量を求める関数 $f(p, d)$ を定義する。すると、アプリケーション p の消費電力 E_p は式 5 で表される。

$$E_p = \sum_{i=1}^{n} f(p, d_i) c_i \tag{5}$$

ここで、OS や待機電力を特殊なアプリケーションとして抽象化すると、デバイス全体の消費電力は各アプリケーションの消費電力の和となる。そこで、式 4 と式 5 から、式 6 が導き出される。

$$E = \sum_{i=1}^{m} \sum_{j=1}^{n} f(p_i, d_j) c_j \tag{6}$$

pSurvive では、この式に従って消費電力を計測する。そのために必要な情報は $f(p, d)$ 及び c、つまりアプリケーションがどのデバイスをどれだけ使っているのかという使用量と各デバイスの単位使用量あたりの消費電力量である。よって、アプリケーション別の消費電力計測ではこの二つを計測する。

2.1.1 アプリケーション別消費電力量の計測

まず、アプリケーションのデバイス使用量を計測する方法については、アプリケーションはデバイスを使用する際に必ずデバイスドライバを通してアクセスを行うため、デバイスドライバに機能を追加するという手法を用いる。また、CPU やメモリについてはデバイスドライバに相当するものが無いため、OS の統計情報などからデータを取得する。デバイスの単位使用量あたりの消費電力量 c については前もってキャリブレーションを行い、実測値から求めることで対応する。

2.1.2 将来の消費電力量予測

消費電力の計測時に消費電力量の合計値だけでなく、電力の消費履歴を記録しておくことで、時系列に過去の消費電力履歴を参照する機能を持つ。これにより、消費電力が常に一定とならないアプリケーションについても動的に予測を立てることが可能となる。

2.2 アプリケーション動作時間保証機構

本節では、前節の消費電力予測を元に、どのようにしてアプリケーションの動作時間保証を実現するかについて述べる。

2.2.1 保証対象となるアプリケーション

保証対象アプリケーションには、利用者やアプリケーション側から任意に設定可能なものの他に、システムを最低限維持するために必要なものがある。前者は、利用者が明示的に指定するか、アプリケーション側から保証要求を行うことで保証対象となる。アプリケーション側からの要求とは、サービス提供者側がアプリケーションの一定の動作時間を保証したい場合に行われる。例えば、子供の見守りアプリケーションなどはそのサービスを実現するために最低限 1 日程度の動作を保証する必要があるとすると位置情報を知るための GPS センサ、位置情報を送信するための無線通信、さらに後述するシステムを最低限動作するための電力を 1 日分予約することになる。

システムに最低限必要なものとしては、OS そのものや待機電力といったものが挙げられる。これらはノードの電源を入れている以上必ず必要になるものなので、常に考慮しておく必要がある。そのため、これらはシステム維持のための一つのアプリケーションとして抽象化する。これにより、他の一般アプリケーションと同じ枠組みで消費電力管理が可能となり、システムが単純化される。

2.2.2 予約電力量モデル

まず始めに、保証対象となるアプリケーションを $R_1, R_2, R_3, ..., R_k$ し、それぞれに要求された保証時間を $T_1, T_2, T_3, ..., T_k$ とする。すると、アプリケーション R を T だけ動作するために必要な電力量 E_R は式 5 から式 7 で表される。

$$E_R = T \sum_{i=1}^{n} f(R, d_i) c_i \tag{7}$$

よって、全ての保証対象アプリケーションが必要な電力量 $E_{required}$ は、式 6 を用いて式 8 で表される。

$$E_{required} = \sum_{i=1}^{k} (T_k \sum_{j=1}^{n} f(R_i, d_j) c_j) \tag{8}$$

この式から、pSurvive では $E_{required}$ だけの電力量を保証対象アプリケーションのために残しておくことで、動作時間保証を実現できることが分かる。

2.2.3 動作時間保証の実現方法

pSurvive では、動作保証を実現するために動作保証対象以外のアプリケーションの動作を抑制する。具体的にはバッテリ残量が $E_{required}$ になった時点を動作保証対象以外のアプリケーションにとっての電池切れと考え、制御を行う。制御の方法にはいくつかの種類が考えられる。

まず最も簡単なものは単純にアプリケーションを終了させるというものである。この手法はアプリケーションにとっては突然終了させられてしまうので柔軟性は無いが、実装が容易である。また、終了するという動作は通常の電池切れの場合と同様であるので、この手法は既存アプリケーションに手を加えずに採用する事が可能である。

次に、終了するのではなくアプリケーションから見た見かけ上の電池残量を少なく見せることにより、アプリケーションに省電力モードなどの機能 (フィデリティ調整機能) があればそれを用いるように促すという手法がある。この手法は、既にアプリケーション側に省電力のための機構が実装されていれば有効だが、そうでないアプリケーションに対しては新たに実装する必要があるため、実装のための開発コストが必要になる。

最後に、アプリケーションではなく OS 側から各アプリケーションの資源利用を抑制することで、単位時間あたりの消費電力を減らして延命を図るという手法がある。この手法は、アプリケーションに変更を加えることなく適用可能であり、効果が高いように思われるが、アプリケーションの資源割り当てを OS が抑制する機構を実装するには、既存のリアルタイム OS の枠組みにまで踏み込んだ議論が必要となり、システムの複雑性が増してしまう。

以上の手法の中から、pSurvive では問題をアプリケーションの動作時間保証に限定するために、単純にアプリケーションを終了させる手法を選んだ。資源の効率的な利用という観点からはその他の手法を選ぶ方が望ましいが、本手法が最も既存アプリケーションの修正が必要がないため、汎用性の高さという点でメリットがある。

3 実装

本節では、pSurvive システムの実装について述べる。ARM バージョンは組み込み型センサノードを想定し、多数あるセンサデバイスを細かく制御することで pSurvive のモデル正統性を検証するために利用した。一方、Android バージョンは、より一般的なデバイス向けに実装することで、スマートフォンや Android タブレット端末などの昨今実際に普及している端末での評価を行うために利用した。

3.1 ARM バージョン

以下に示す要件に基づいて、Linux カーネル 2.6.21(ARM) 上でマイクロユビキタスノード上での実装を行った。

1. **OS に手を加えることが可能で、開発環境が利用できること**
   アプリケーションの各デバイス使用量モニタリングを OS レベルで実装する必要がある。

2. **実際にマイクロユビキタスノードでの採用例があるか、製品としてすでに販売されていること**
   実際の組み込みシステムでの採用例があることで、実験結果の信憑性が高まる。

3. **本システムの有効性を示すのに十分な評価が取れるだけの機能を有すること**
   細かな評価を行えるよう、既存のマイクロユビキタスノードと同等かそれ以上の機能を持っていることが望ましい。

4. **特殊なハードウェアを極力用いず、本システムの汎用性を示せること**
   ハードウェア依存の実装になりがちな組み込みシステムの中で、なるべく特定のハードウェアに依存するところを無くし、実験環境以外のマイクロユビキタスノードでも同じことが言えるということを保証する。

本研究では、前述した要件を満たすマイクロユビキタスノードとして、Gumstix Vertex XM4-BT 及び NUTS センサボードを用いた。

Gumstix Vertex XM4-BT は、Gumstix 社が販売している ARM CPU を搭載した組み込みハードウェア製品である。Linux カーネル 2.6.21 が動作しており、arvell 社の Xscale PXA270 を CPU として搭載している。Xscale は PDA などで広く使用されている組み込み用 CPU であり、マイクロユビキタスノードとしての利用実績が豊富である。また、広く普及している ARM プロセッサであるため、既に多くのアプリケーションが公開されていることも選定の理由である。詳しいハードウェア仕様を表 3 に示す。

NUTS センサボードは、本研究グループが開発した、Gumstix Vertex の拡張コネクタに接続して使用できるセンサボードである。Gumstix の拡張デバイスとして働き、音声入出力、照度、加速度センサ、GPS、ロータリーエンコーダといったセンサや液晶ディスプレイ、USB ポートを介して各種機器を搭載できる。PMU(Power Management Unit) を介した電池残量計測と、各デバイスへの電力供給制御を行える。

3.2 Android バージョン

以下に示す要件に基づいて、Android バージョンの実装を行った。

1. **ファームウェアの書き換えが必要でないこと**
   カスタムカーネルによるファームウェアの書き換えなどを行わずに、Android OS に搭載されている Linux の機能を元に実装する。これにより、既に製品として多く販売されている Android 端末上でソフトウェアをインストールするだけでの利用が可能となる。

2. **製品としてすでに販売されている端末上で動作すること**
   実際に販売されている製品上で動作できることで、本実装の汎用性を示すことができる。

図 8: Gumstix Vertex XM4-BT

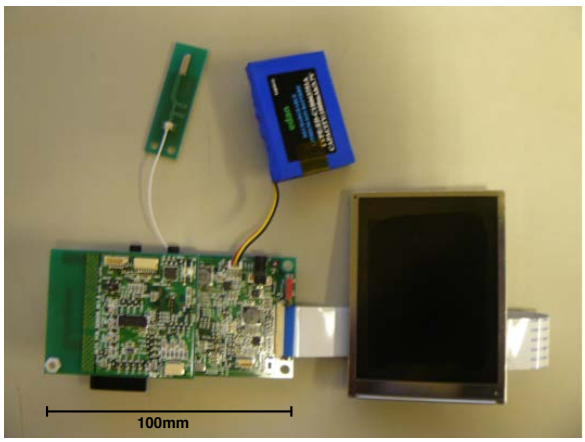

図 9: NUTS センサボード

3. **本システムの有効性を示すのに十分な評価が取れるだけの機能を有すること**
   一部の Android 搭載機種では、ベンダによるカスタマイズによってアプリケーションの動作に制限が加えられている場合があるが、そうした端末はサポートせず、最も一般的な端末を対象とする。

本研究では、前述した要件を満たす Android 端末として、Google Nexus One を用いた。

Google Nexus One は、HTC 社が製造し、Google が販売する Android 開発者のためのリファレンス機である。Android 本来のもつ機能に対して特殊な拡張や制限が加えられていないため、本端末向けに実装したプログラムは他の多くの Android 端末でも動作させることができる。また、WiFi や GPS といった昨今の Android 端末の多くが有する機能を持っているため、実装環境として適切であると判断した。

4 pSurvive の評価

本節では、pSurvive システムを用いた実験を行い、ARM バージョンの成果を示す。評価はアプリケーション別消費電力計測・予測機構とアプリケーション別動作時間保証機構の各機能別のものと、システム全体の評価について行い、pSurvive の有効性について考察する。

4.1 実験環境

まず最初に、実験環境について解説する。リチウムイオンバッテリは常に線形な電圧低下を行うわけではない。そのため、本実験ではバッテリ特性による測定結果への影響を避けるため、減衰傾向が線形となる電圧区間のみを有効な残バッテリ量として扱う。また、いくつかのバッテリを用いて実験したところ、固体によって満充電時の電圧値に開きがあることがわかった。こうした個体差が測定結果に影響を及ぼさないようにするため、一連の実験は全て一つの同じ端末において、同じバッテリを用いて行った。

消費電力計測機能の評価のために、まずはデバイスの単位使用量あたりの消費電力を計測する。本実験では、実験環境の動作状態を最低動作環境と、それに加えて CPU、無線 LAN を使用しているそれぞれの状態についてバッテリ使用量を計測した。この際、CPU の負荷発生にはモンテカルロ法による円周率計算、無線 LAN の通信には 255 バイトの UDP パケットを送信し続けるというプログラムを使用した。それぞれの動作状態において、各デバイスの使用状況は以下の様になる。

4.2 デバイスの単位使用量あたり消費電力の計測

- 最低動作環境（A）：最低限の動作環境
- CPU のみ使用環境（B）：A + CPU 負荷

表 3: Gumstix Vertex XM4-BT の仕様

CPU	Marvell PXA 270 with XScale, 400MHz
RAM	64MB
Flash	16MB
拡張コネクタ	60Pin ヒロセコネクタ, 80Pin コネクタ、24Pin Flex コネクタ
大きさ	80mm x 20mm x 6.3mm
重さ	8g
動作電圧	DC 3.6V - 5.0V
バッテリ	edan ED-C1B063751A, 1000mAh/3.7V

表 4: 各デバイスの単位使用量辺りの消費電力

デバイス	最低動作環境	CPU	無線 LAN 待機	無線 LAN 通信
1 単位	60 秒	1 秒	60 秒	10000 Packets
単位消費電力（Units）	18.28	1.15	53.09	5.99

- 無線 LAN 電源のみ ON 環境（C）：A + 無線 LAN 待機電力
- 無線 LAN 通信環境（D）：A + 無線 LAN 待機電力 + 無線 LAN 通信電力

このことから、各デバイスの消費電力は以下の式で表すことができる。

- CPU：B - A
- 無線 LAN 待機電力：C - A
- 無線 LAN 通信電力：D - C

この式を元に、各デバイスの単位消費電力を計測した結果が表 4 である。

ここで、最低動作環境、及び無線 LAN 待機電力は実際の計測時間とバッテリ消費量から算出した値である。また、CPU 使用量は Linux の/proc/[Process ID]/stat から取得できるプロセスのユーザーモードでの実行時間とカーネルモードでの実行時間の和を用いた。そのため、データ取得時は jiffies と呼ばれる時間単位で取得でき、その粒度は 100 分の 1 秒である。無線 LAN 通信については、本実験においては送信のみを対象とし、ifconfig コマンドから取得できる送信パケット数を参照した。通信における単位使用量については、パケット数の他にデータ量によるものも考えられたため、255 バイトのパケットを用いた場合と 511 バイトのパケットを用いた場合での電力使用量の違いも測定したが、結果電力使用量の変化は送信パケット数について線形に変化していく傾向が見られたため、パケット数を単位として用いた。

4.3 アプリケーション別消費電力計測機構の評価

次に、計測した単位使用量を元にして、計算で求めた消費電力と実際の消費電力を比較することで、消費電力計測の精度を求める。前節のネットワーク計測のプログラムと円周率計算の両方の処理を行うプログラムを作成し、二つの処理を同時に行った場合の CPU とネットワークの使用量を計測する。実験では 10 分間の計測を行い、前後 1 分間を計測時におけるタイミングのずれを排除するために切り捨

表 5: 複数デバイス使用時における消費電力予測

デバイス	最低動作環境	CPU	無線 LAN 待機	無線 LAN 通信
1 単位	60 秒	1 秒	60 秒	10000 Packets
単位消費電力（Units）	18.28	1.15	53.09	5.99
計測した使用量	480 秒	358.5 秒	480 秒	372768 Packets
消費電力（Units）	146.27	411.70	424.75	223.29

て、8 分間分のデータを有効データとして扱った。実際に計測したデバイス使用量と 5.3 で計測した単位消費電力から求められた合計消費電力をまとめたものが表 5 である。

この表における値は理論値であり、ノード全体の予想電力使用量 $E_{estimated}$ は各デバイスの消費電力の和であるから、式 9 で表される。

$$\begin{aligned} E_{estimated} &= 146.27 + 411.70 + 424.75 + 223.29 \\ &= 1206.01 \end{aligned}$$

一方、バッテリから取得できる消費電力量の実測値 E_{real} はバッテリから取得でき、その値は 1331 であった。ここで、実測値と理論値の間の誤差は式 9 で表され、

$$\frac{|E_{real} - E_{estimated}|}{E_{real}} \times 100 = 9.39$$

結果、9.39%となる。この誤差が発生した原因としては、いくつかの原因が予想される。第一に 5.1 でも触れたバッテリのセンシングにおける揺らぎによるもの、第二には 5.3 におけるデバイスの単位消費電力の測定誤差によるもの、第三は各デバイスを区分けする際に無視してしまった要素によるものがあるだろう。無視してしまった要素とは、無線 LAN の受信電力であったり、OS オーバヘッドの揺らぎによるものなどである。このような誤差の取り扱いについては、動作保証機構において誤差分の余裕を持ってバッテリを確保することによって問題が解決できると考えられる。

4.4 消費電力量予測の評価

実際にデバイスを継続して動作させた際の消費電力量予測についての実験を行った。実験環境は前節で用いたのと同じ CPU と無線 LAN に負荷をかけたものを 1 時間動作させ、N 分後のバッテリ残量を予測した。予測アルゴリズムには移動平均アルゴリズムを用いたが、本実験においてはデバイスの使用状態は実行中ほぼ一定であるため、予測アルゴリズムの違いによる精度の違いは無視できる。実験結果におけるバッテリ残量と予測値の誤差をプロットしたものが図 10 である。

図は 1 分、5 分、10 分、20 分、30 分先の消費電力量を予測し、その予測値が実測値とどれくらい離れているかを表している。結果から予測時間が長くなれば長くなるほど誤差が大きくなる傾向があることが分かった。実際に消費電力予測を利用する際は、こうした予測時間による誤差も考慮する必要があると考えられる。

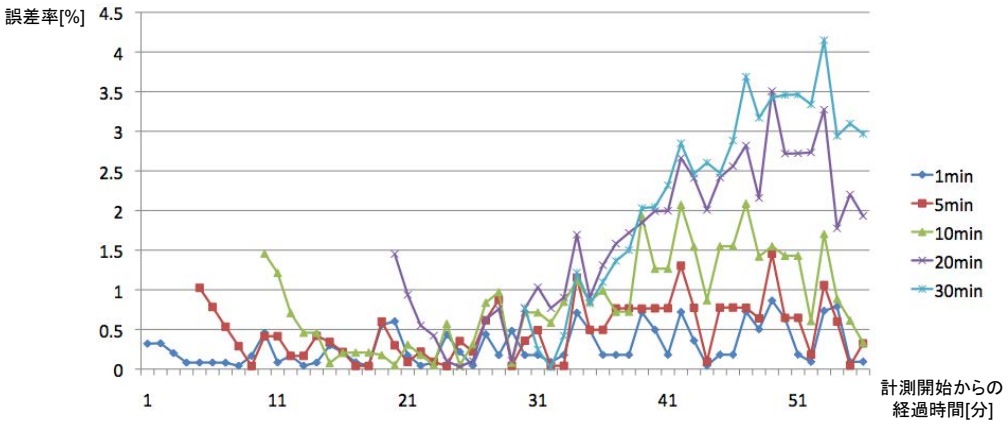

図 10: 予測期間による誤差の違い

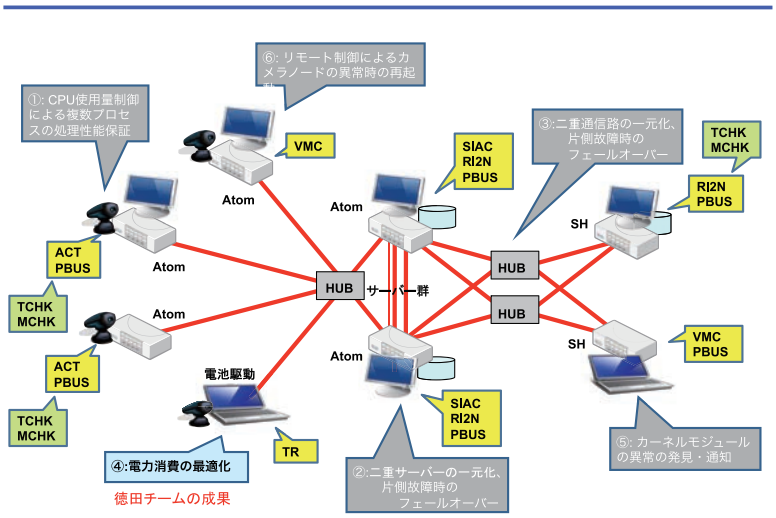

図 11: 統合デモにおけるディペンダブル消費電力管理機構の位置づけ

② 研究成果の今後、期待される効果

ディペンダブル消費電力管理機構は、以下の 2 つの観点において組み込み機器のディペンダビリティ支援に貢献する。まず本機構は、消費可能な有限の電力において、組み込み機器の所定時間の動作 (**アベイラビリティ**) を保証する。これは、電力資源の **failure-proof** である。電池駆動機器においては、全電池容量に対してアベイラビリティ保証の時間を指定できる。AC 駆動機器においては、所定の電力消費量に対して同様のことを行える。すなわち、電力消費の最適化が可能となる。これは、電力資源の **performance-degradation-proof** である。また本機構は、組み込み機器の開発フェーズ、運用フェーズ、および保守・更新フェーズのディペンダビリティ支援を行うことを目的とする。**開発フェーズ**においては、開発対象機器上で動作しうるプロセスのリストが判明しているとき、それらのプロセスを一定時間、一定負荷未満で動作させられる電力量を見積もるツールを検討する。**運用フェーズ**においては、電源電力の監視とプロセスの選択的停止により、アベイラビリティ支援を行う。**保守・更新フェーズ**においては、開発時に見積もった挙動と、実際の挙動との差異をログから抽出するツールを検討する。同ツールを用いることにより、電源の劣化を検出し、ハードウエアフォルトを早期に発見可能となることが期待できる。

次に、本研究の新規性ならびに画期的な点は以下の通りである。ここでは関連研究との比較において論じる。まずプロセス毎の資源管理に関して、宮川らの提案する POPM は、昨今の CPU が備える DVFS (Dynamic Voltage and Frequency Scaling) 機能を使い、プロセス毎に必要十分な CPU 資源の割り当てを行う [13][14] ことで消費電力を削減する。POPM では CPU の資源管理にのみ着目しているため、CPU 以外の資源については特に対応していない。これに対してディペンダブル消費電力管理機構では、CPU 以外の資源の消費電力をプロセスごとに計測し、将来の消費電力を予測することで、特定のアプリケーションを対象とした**指定時間の電力アベイラビリティ保証を実現**する。TinyOS は資源の限られたセンサノードに対して、nesC と呼ばれる C の拡張言語を用いてプログラミングを行う [15]。nesC の特徴はプログラムをコンポーネントと呼ばれる小さな機能単位に分割し、それらのコンポーネントを接続するだけで基本的なアプリケーションを容易に作成できるところにある。各コンポーネントは必要な時にのみ有効にされるため、各デバイスは必要なときにだけ起動される。こうした研究は、デバイス毎に電源を制御することでバッテリ管理を実現するという点において本研究との関連がある。しかし、一般的な組込機器では nesC のような特殊なプログラミング言語を用いることは不可能である。本機構は、既存アプリケーションの変更が不要であるため、より**幅広い製品での応用が可能**である。

次に、消費電力計測・予測に関して、Xiaofan, J. らはセンサノードに取り付けることで、デバイスの消費電力量を詳細に計測できるハードウェアを開発した [16]。この手法では、既存のセンサノードに提案するマイクロパワーメータを実装する必要があり、研究・開発における実験環境であればまだしも、全ての製品に搭載するには 1 で挙げたような搭載コストを始めとする問題が残る。この問題を受けて、Dunkels, A. らはセンサノードにおいてソフトウェアでの使用電力予測手法を提案している [17]。この研究で採用しているモデルは本研究とほぼ同じもので、各デバイスの使用時間を測定し、単位時間あたりの使用電力をかけあわせたものをノード全体の消費電力とする。Dunkels, A. らの研究では各デバイスの単位時間あたりの使用電力を計測するのにデジタルオシロスコープを用いて直接回路上での電力を測定している。この手法は、実際のハードウェアの消費電力を物理的に測定するので現実の値との誤差がほとんど無いデータを取得することができる。一方で、回路に直接プローブを接続するという計測手法は、近年のマイクロユビキタスノードの小型化に伴い実現が難しくなっている。例えば、1 チップ内に CPU、無線通信、センサなどの機能を全て搭載した LSI などの場合、そもそも機能別の電力を計るためのプローブを接続することができないため実現不可能である。

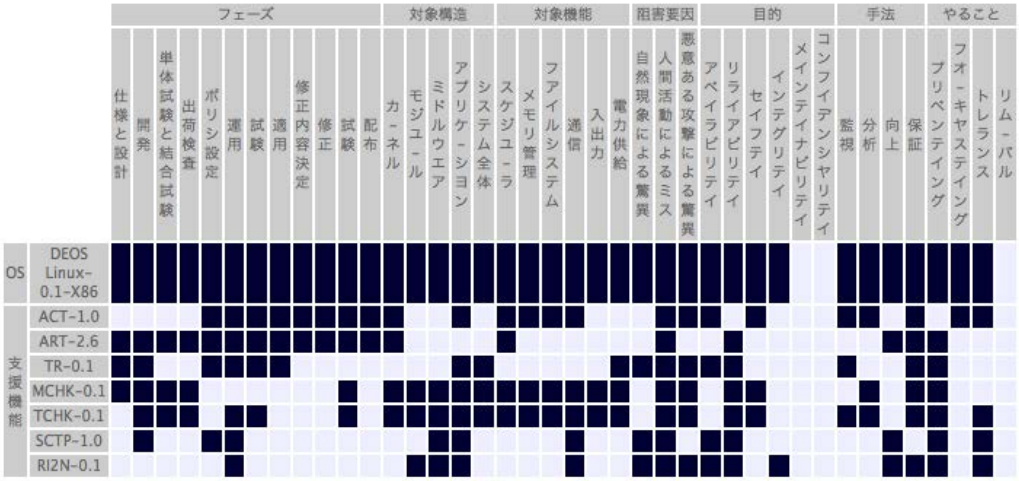

図 12: ディペンダビリティ支援マップ

(1)-3 D-Case およびディペンダビリティメトリクス

① 研究実施内容及び成果

平成 20 年度より、任意の組み込み機器が提供するディペンダビリティの量と質を数的に評価できることを目的として、メトリクスの開発を行っている。当初の検討結果では、質を表すために、複数の評価軸を組み合わせる必要があると考えた。たとえば以下は軸の例として挙げられる。

*-proof	energy	performance	threat	fault	
ディペンダビリティ属性	availability	reliability	integrity	maintainability	confidentiality
ディペンダビリティ向上手段	fault prevention	fault removal	fault forecasting	fault tolerance	
PDCA	plan	do	check	act	
マルチレベル支援	level1	level2	single core	multi core	multi node

これらのうち、たとえば PDCA 軸と*-proof 軸を組み合わせた平面でディペンダビリティの質を議論することを想定する。その場合は、plan 時の performance 保証を目的とした機能の数とそれ自体の性能や、do 時の fault からの回復を目的とした機能の数とその性能などを分析していくことになる。そこで上記の軸を用い、ディペンダブル OS 全体をディペンダビリティ支援機能の集合と捉え、それらを評価できる Web サービスを平成 20 年度に構築した。図 12 および図 13 に、同サービスを用いて導出できる評価結果の例を示す。

これらの評価では、開発者が上述の Web サービスを用いて開発中システムが満たすディペンダビリティ支援を自己申告し、そのシステムが満たすディペンダビリティ支援の内容を直感的に図として示すことを意図していた。このため、開発者が申告したディペンダビリティ属性を、そのシステムが本当に満たすことを証明する手段が必要となった。そこで、ディペンダビリティケース (D-Case) と呼ばれる手法によりその証拠 (Evidence) を明らかにできるようにした。また、D-Case を定量化することで、システム全体のディペンダビリティ評価を可能とした。以下にそれらを詳述する。

1 D-Case およびメトリクスの目的と手法

組み込み機器に搭載されるオペレーティングシステムのディペンダビリティ支援を質的および量的に議論するための基準 (以降、ディペンダビリティメトリクスあるいは単にメトリクスと呼ぶ。) を提案する。同メトリクスの目的は次の通りである。まず、オペレーティングシステムの開発者や利用者、あるい

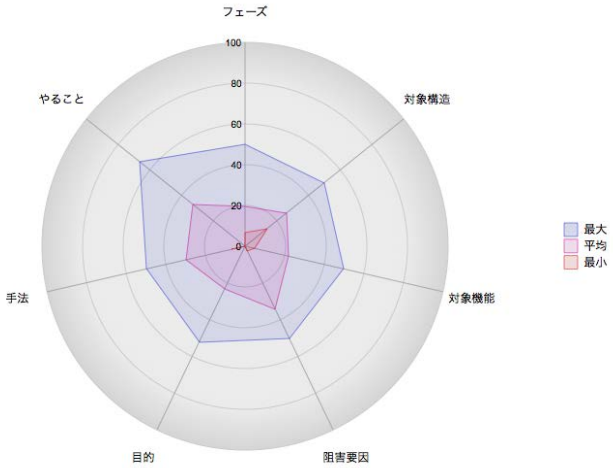

図 13: ディペンダビリティ支援チャート

は評価者は、本メトリクスを用いて、そのオペレーティングシステムが提供できるディペンダビリティ支援を表明可能でなければならない。次に組み込み機器の開発者は、開発対象の機器が必要とするディペンダビリティ支援を見積可能でなければならない。開発者はまた、本メトリクスを用いて、採用候補オペレーティングシステムの比較検討を行える必要がある。最後に、組み込み機器の利用者は、類似する複数の組み込み機器が具有するディペンダビリティを、本メトリクスを用いて比較可能でなければならない。以上より本メトリクスは、オペレーティングシステム開発者、組み込み機器開発者、および組み込み機器利用者による利用を対象とする。

本メトリクスは、組み込み機器の開発者を念頭に置き、単に機器の実行時だけでなく、機器のライフサイクルにおける開発時や保守更新時に提供されるディペンダビリティ支援を含めて議論可能とする。従って、単にオペレーティングシステムカーネルやカーネルモジュールのような実行時の挙動が重要となる機構だけでなく、製品の開発時や保守更新時に重要となる検証、デバッグ、あるいは解析ツールなども、本メトリクスによる評価対象に含む。また、組み込み機器の利用者を念頭に置き、複数の機器を粗粒度の指標で比較できるようにする。これは、H、HB、あるいは B のように表現される鉛筆の濃さや、5 段階や 10 段階評価の学業成績の発想に類似している。

本メトリクスは、以下に示す手法により、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支援を網羅的に評価可能とする。図 14 に概念を示す。まず、オペレーティングシステムが提供する機構を、ディペンダビリティ支援を行うものとそうでないものの 2 つに分類する。オペレーティングシステム全体のディペンダビリティは、ディペンダビリティ支援を行う各機能を個別に評価した値を合算したものとする。その際、時間、空間、あるいは程度など様々な軸を設けて、評価対象機構が提供するディペンダビリティ支援の内容を定性的に評価する。また実システムにおけるディペンダビリティ要求の見積時には、同様の軸を用いて要求を定性的に評価する。ただし定性評価だけではオペレーティングシステムが、いずれかの属性を実際に満たしているかどうかや、実システムにおいて求められる具体的な性能指標等を扱えない。

そこで本メトリクスでは、ベンチマークやフォルトインジェクション、検証技術等による具体的な値により定性評価結果を補足する。オペレーティングシステム開発者は、GSN[18] を用いて定性評価軸に含まれる項目と具体的な値との関連づけグラフを構築する。このグラフを D-Case(Dependability Case) と呼ぶ。

次にオペレーティングシステム開発者は D-Case からディペンダビリティスコアを導出 (数値化) し、それを上述した評価軸から選択した 2 軸の平面、あるいは同様に 3 軸の空間に可視化する。ディペンダ

表 6: 定性評価項目

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	

ビリティスコアは、任意の粒度のディペンダビリティ支援機構が提供できるディペンダビリティ支援の最大値である。その粒度として、各支援機構、複数の支援機構の組み合わせ (コンフィグレーション)、あるいはオペレーティングシステム全体を考えることができる。システム開発者は、これらのスコアと、それを可視化した情報により、複数の異なる機構やオペレーティングシステム同士の比較が可能となる。

これに加えてシステム開発者は、ディペンダビリティ要求見積結果を D-Case を用いて表現し、それをオペレーティングシステムごとの D-Case と比較することができる。これにより、それらのオペレーティングシステムを用いてシステムを開発した際に満たされるディペンダビリティ要求と、それらを用いる際のコンフィグレーション (ディペンダビリティ支援機構の組み合わせ) を求めることができる。

2 ディペンダビリティの定性評価

ディペンダビリティは多様な性質の集合であることから、本文書では、ディペンダビリティに関する評価や見積りを行うための定性評価方式をまず提案する。オペレーティングシステム開発者は、ディペンダビリティ支援ツールや機構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。例えば、あるディペンダビリティ支援機構が可用性を向上させると考えられるとき、その機構の定性評価項目のひとつとして、可用性をチェックする。次に本文書では、Goal Structuring Notification(GSN) をディペンダビリティ評価に特化させた、Dependability Case(D-Case) を提案する。D-Case を用いて、オペレーティングシステムの開発者は、そのオペレーティングシステムの定性評価結果に対し、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、定性評価結果の具体的根拠を与えられる。システム開発者は、ディペンダビリティ要求として、あるベンチマーク結果がある閾値未満であることなどの証拠を要求できる。

定性評価は、ディペンダビリティ支援機構やツールの効果を、表 6 に示す時間、空間、対象など複数の軸で評価するものである。以下の各項で、それらの軸を詳述する。

2.1 阻害要因

まず、ディペンダビリティ支援機構が扱う問題を明らかにすることが重要である。たとえばあるオペレーティングシステムが、温度や湿度など自然環境の急激な変化に適応して挙動を変更し、システムの不具合を避ける機構を有する場合とそうでない場合とでは、提供されるディペンダビリティの質が異なるといえる。従って本メトリクスでは、ディペンダビリティ支援機構が効果を発揮する原因を、ディペンダビリティ阻害要因という視点で捉えることとする。阻害要因は、組み込み機器の内部に存在するものと、外部に存在するものとに分類できる。内部要因には、人的ミスにより混入したバグや、ハードウエアの故障が含まれる。外部要因には、自然現象による脅威や悪意ある攻撃による脅威が含まれる。これらをより詳細に分類することも可能ではあるが、メトリクスの煩雑化を避けるためこれ以上の詳細化を行わない。以上より、本メトリクスでは、表 6 に示す事項を阻害要因として扱うこととする。

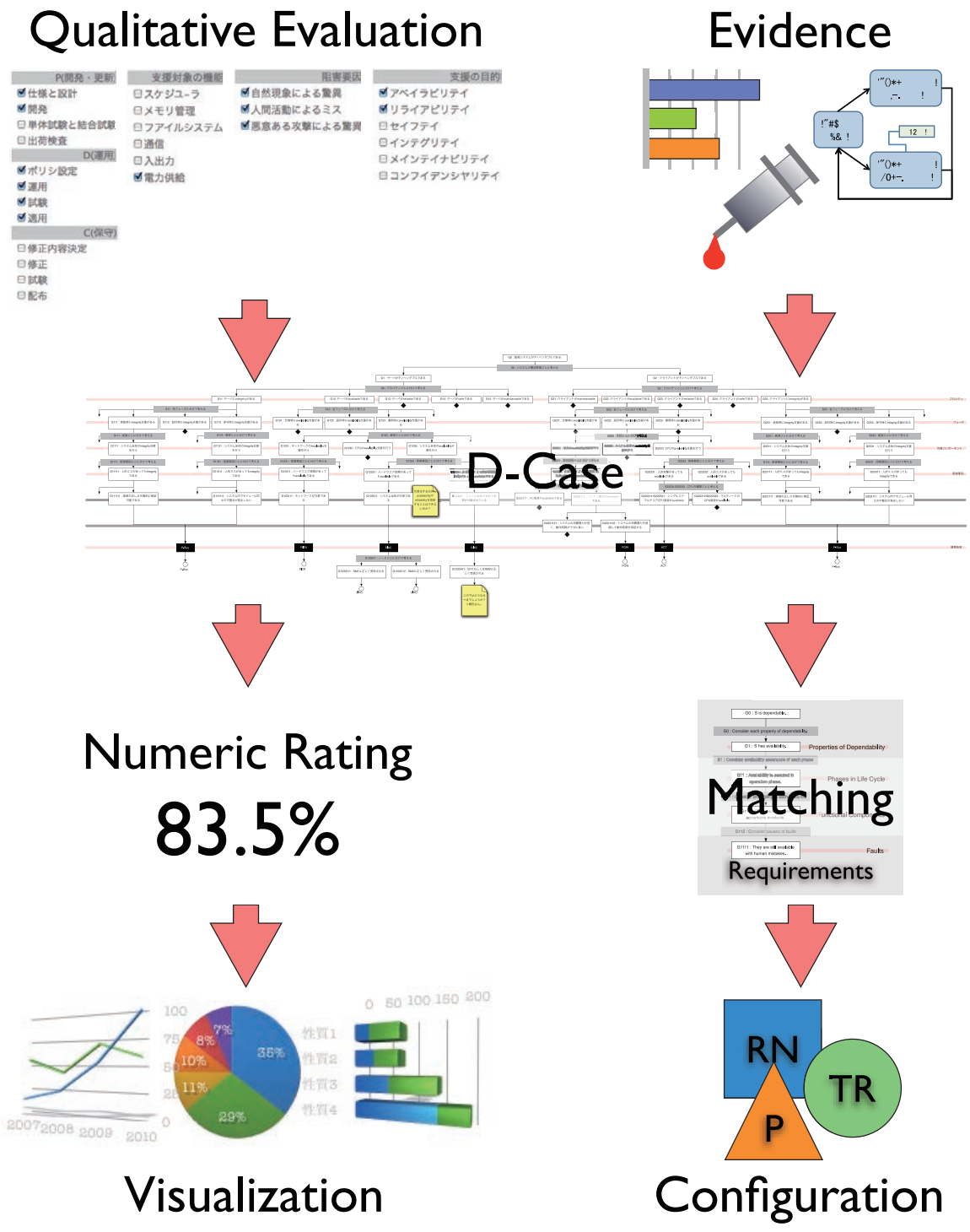

Figure 14: Concept

図 14: コンセプト

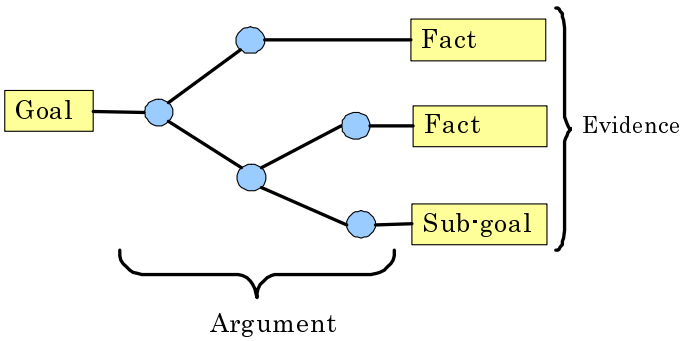

図 15: Assurance Case の枠組み

2.2 目的性質

一般的に、可用性や信頼性といった概念は、ディペンダビリティという概念の一部を構成する性質であると考えられている。異なる構造や機能を対象とするディペンダビリティ支援機構同士が、同一の目的を (たとえば信頼性の向上など) を共有することがある。また、同一の構造を対象としながら、可用性や安全性など、目的が異なるものもあると考えられる。それらの機構同士は、相互補完的な関係である。このような関係を発見し、検討する上で、ディペンダビリティ支援機構の目的を明らかにすることは重要である。本メトリクスでは、表 6 に示す事項により、ディペンダビリティ支援機構の目的を分類する。これらはディペンダブルシステムに関する論文において広く参照されている概念 [19] である。

2.3 フェーズ

ディペンダビリティ支援やツールには、単にシステムの実行時だけではなく、開発時や保守・更新時に効果を発揮するものが存在する。システムのライフサイクルにおけるこれらの段階を、本稿ではフェーズと呼ぶ。また、システムやそれを構成する機器の開発自体にも、設計、実装、検査などの作業があるように、各フェーズはより細かな段階に分割して考えることができる。効果を発揮するフェーズが異なるディペンダビリティ支援機構同士を同列に議論することはできない。以上より本メトリクスでは、表 6 に示す項目により、ディペンダビリティ支援機構が有効に機能するフェーズを示す。

2.4 対象機能

ディペンダビリティ支援機構が支援の対象とする、オペレーティングシステム内の機能は多岐にわたる。実時間性を向上させてシステムの応答性を高め、結果として各時刻におけるシステムの可用性や信頼性を向上させる機構は、スケジューリング機能を対象とした機構であると言える。また、アプリケーションプログラムに対して、ネットワーク通信における一定の帯域を保証して、ネットワークの信頼性を向上させる機構は、通信機能を対象とした機構であると言える。異なる機能を対象とした複数のディペンダビリティ支援機構を同列に議論することはできない。そこで本メトリクスでは、ディペンダビリティ支援機構が効果を発揮する対象の機能を明確にする。一つの OS には非常に多数の機能が含まれているが、本メトリクスでは当面、表 6 に示す機能を支援の対象として扱うこととする。

3 D-Case による証拠付け

本メトリクスでは、D-Case と呼ぶ枠組みを用いて、定性評価による議論をより客観的にする。

定性評価項目は、それがオペレーティングシステムのディペンダビリティ評価に用いられた場合には、あるディペンダビリティ支援機構やツールがそれぞれの項目を満たすかどうかという開発者の主観によっ

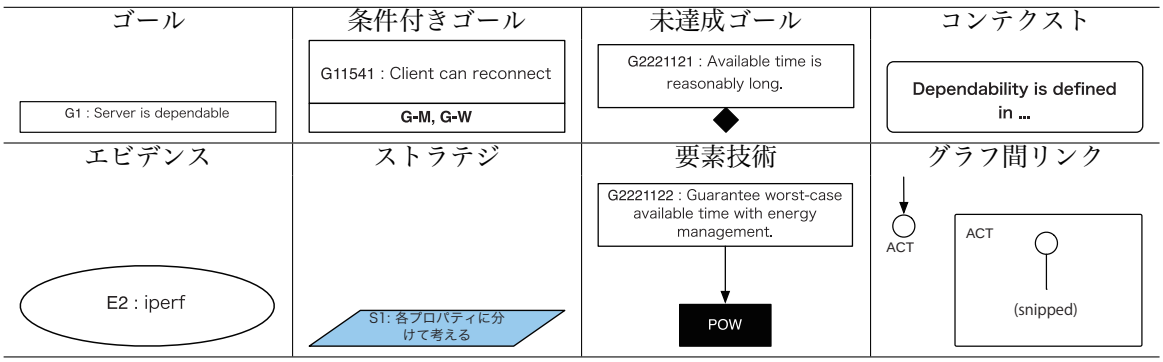

図 16: 本メトリクスにおける GSN 表記法

て、またシステム開発者がディペンダビリティ要求見積りに用いる場合には、その項目が必要かどうかという主観によって、チェックの有無が決められる。本メトリクスでは、これらの主観に対してベンチマークや検証、フォルトインジェクションによる結果を、主観による評価の証拠あるいは見積りの具体値として対応付ける。これにより、客観的かつ具体的な評価や見積もりを可能とする。本メトリクスは、この対応付けを Dependability Case(D-Case) により可能とする。D-Case は Assurance Case[20] と呼ばれる枠組みの応用である。図 15 に Assurance Case の枠組みを示す。Assurance Case では、ステークホルダへ提示したい事項を議論の枠組みを構築しながらサブゴールに分割していき、実験や公理等から得られるエビデンスによってそれらを保証する。本メトリクスでは Assurance Case の枠組みを用いて、ディペンダビリティ支援機構やツールに対する定性評価・見積り結果にベンチマーク、検証、フォルトインジェクション等の結果を事実として対応させる。これにより、評価や見積りの内容をより客観的に議論可能とする。

本メトリクスでは、*Goal Structuring Notation (GSN)*[18] を用いて D-Case を記述する。図 16 に、本メトリクスで採用する表記方法を示す。ゴールは D-Case を用いて示したいことで、例えばオペレーティングシステム A がディペンダブルである、製品 X がディペンダブルである、といった命題である。ストラテジは、ゴールをサブゴールに分けるときの考え方である。要素技術は、あるゴールを実現するときに用いる OS 中の要素技術を表わす。エビデンスは、その直前のゴールを実証するために用いる、実験結果、ベンチマーク（DS-Bench など）、エキスパートの意見などがくる。図 17 に D-Case の記述例を示す。

上述した表記を用いた議論の構成方法 (具体例については次章を参照のこと) について、本メトリクスでは、ゴールの分割法と配置法に関する制約を定める。具体的には、D-Case を 8 レベルの木により構成する。これにより、オペレーティングシステム開発者、システム開発者、およびシステム利用者が同一の構造を持つ D-Case に基づいて議論可能とする。このことは、システム開発者によるオペレーティングシステム間のディペンダビリティ比較や、システムにおけるディペンダビリティ要求とオペレーティングシステムのディペンダビリティ評価のマッチング等に必須である。D-Case における 8 つのレベルとは以下の通りである。

第 1 レベル そのオペレーティングシステムそのものがディペンダブルである、といった最終ゴールで、議論の対象である。

第 2 レベル ディペンダビリティをサブ性質ごとサブゴールに分解して議論する。定性評価項目の「支援の目的」に該当する。

第 3 レベル 第 2 レベルのサブゴールを、システムや製品のライフサイクルにおけるフェーズごとのサブゴールに分割して議論する。定性評価項目の「フェーズ」に該当する。

第 4 レベル 第 3 レベルのサブゴールを、システムを構成する機器に含まれる機能ごとのサブゴールに分割して議論する。定性評価項目の「対象機能」に該当する。

第 5 レベル 第 4 レベルのサブゴールを、ディペンダビリティ阻害要因ごとのサブゴールに分割して議論する。定性評価項目の「阻害要因」に該当する。

第６, 7 レベル 必要に応じて、第 5 レベルのサブゴールをさらに細かいサブゴールに分割して議論する。定性評価項目よりも粒度が細かい。

第 8 レベル 第 7 レベルまでのサブゴールに対してベンチマーク、検証結果、フォルトインジェクション結果等のエビデンスを与える。

これらのうち、第 1 から 5 レベルは必須である。すなわち、OS のディペンダビリティ評価者、ならびにシステムのディペンダビリティ見積者は、第 1 から 5 レベルの項目を上記の方法で分割し、上記の方法で GSN に配置しなければならない。ただし、不要である項目には未達成ゴールノードを付与してそれ以上の議論を中止できる。また、OS 評価者は、第 1 から第 5 レベルに加えて第 8 レベルも必須である。

4 D-Case ツリーの数値化

D-Case に記述されるゴール/サブゴールのうち、エビデンスが付与されたゴールとそうでないものが存在するとき、前者の割合を求めることで D-Case に記述されたゴールの達成率を把握できる。例えば、あるシステムのディペンダビリティ要求が D-Case にすべて記述されていて、かつ、すべてのステークホルダがその記述内容について合意していると仮定すると、その D-Case を上記の方法を定量化した値は、そのシステムのディペンダビリティ達成度合いを示す相対的な指標と見なせる。D-Case ツリーはモニタリングノードにより動作中システムの監視を行えるため、D-Case ツリーの数値化は、開発中システムだけでなく実行中システムのリアルタイムなディペンダビリティ定量化に利用できる。D-Case ツリーの数値化は以下のように行う。今、D-Case ツリー中のゴール G に n 個のサブゴール gi(ただし i は 1 から n) があるものとし、また各サブゴールには重み Wgi を付与できるものとする。このとき、G が満たすディペンダビリティの量を意味する数値 SG を次のように求める。

$$S_G = \frac{\sum_{i=1}^{n}(Wg_i \times Sg_i)}{\sum_{i=1}^{n} Wg_i}$$

ここで Sgi はサブゴール gi の数値である。gi がリーフゴールでありかつエビデンスが付与されている場合、数値は 1 とし、エビデンスがない場合は 0 とする。本式によって算出されるトップゴールの数値がシステム全体のディペンダビリティ評価指標となる。本指標を用いることで、任意の D-Caset ツリーを対象としてリアルタイムなディペンダビリティ定量化を行える。すなわち、D-Case ツリー中の特定のエビデンスが実行時に満たされていないとき、それに起因してシステムのディペンダビリティがどれだけ低下しているか、算出することができる。　一方、本式のみでは異なるシステム同士のディペンダビリティを比較することができない。そこで、異なるシステムを対象として共通に使用することのできる D-Case ツリーを提供する。開発者は共通ツリーを拡張して対象システムの D-Case ツリーを構築することで、当該システムの普遍的なディペンダビリティ評価を実施できる。図 18 にオペレーティングシステムを対象とした普遍的 D-Case ツリーの一部を示す。このような D-Case ツリーをシステムの種別 (自動車システム、バンキングシステム等) ごとに提供することを想定する。

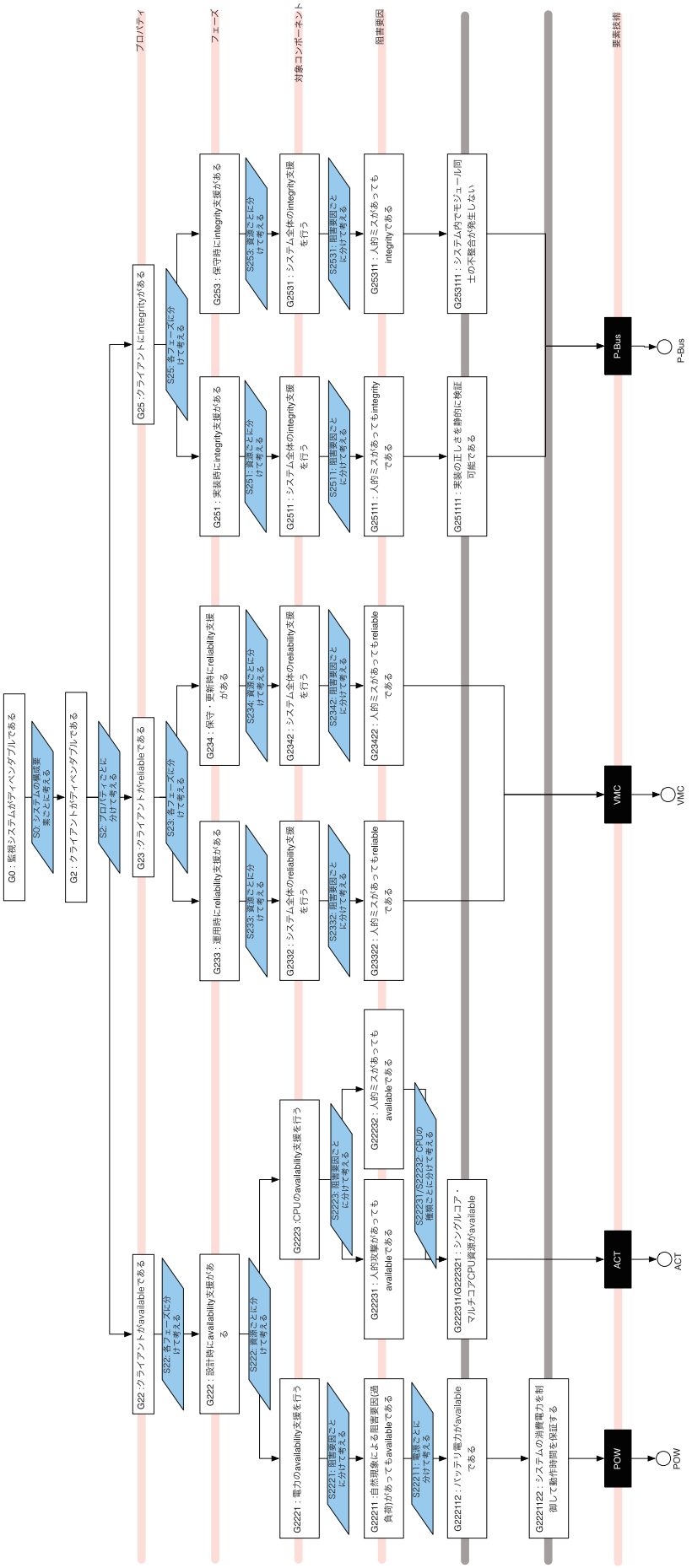

図 17: D-Case 記述例

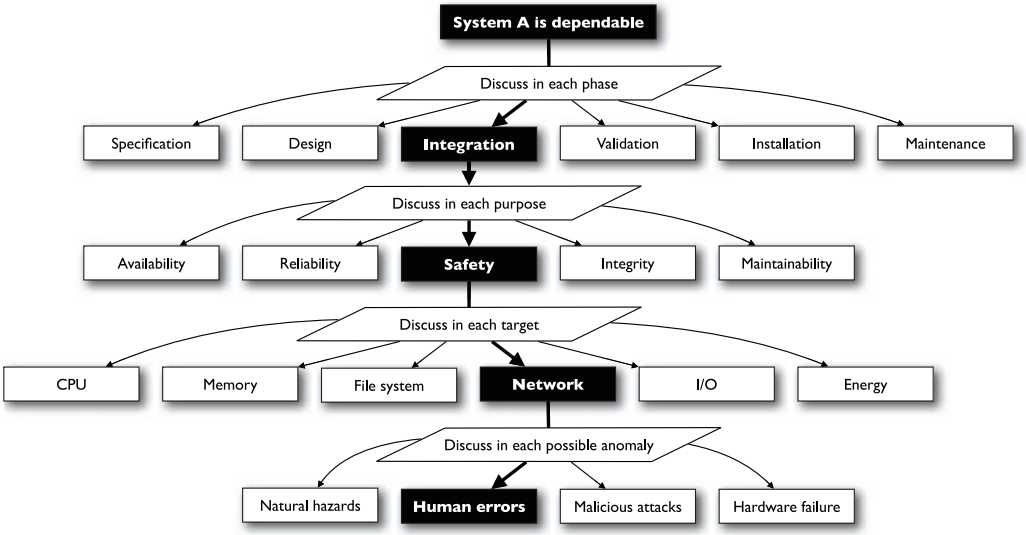

図 18: オペレーティングシステムを対象とした普遍ツリーの一部

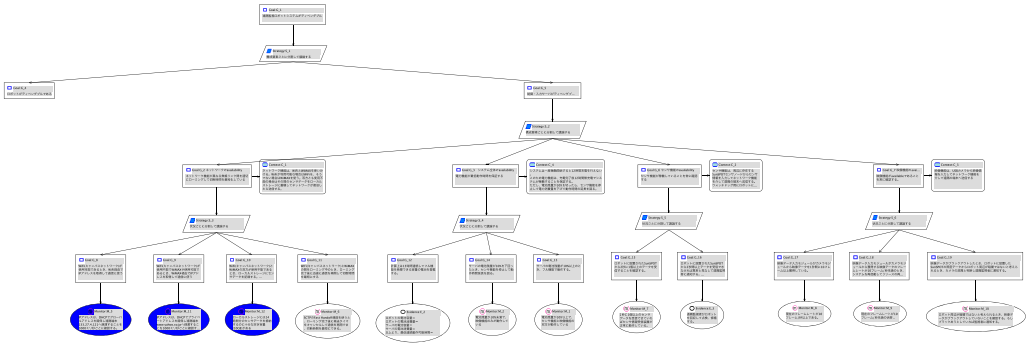

図 19: 動的な実行情報に基づいて D-Case を色付けした例

5　D-Case モニタリングノードの拡張ソフトウエア

D-Case の記述にプログラムを関連づけて、実行中システムに関する情報を動的に取得する機構を構築した。現在まで本研究領域では、ディペンダビリティ実行環境として Linux 上のミドルウエアである D-RE を構築し、それに含まれるインタプリタに D-Case からスクリプトを発行して実行させ、結果として実行情報を得る仕組みが構築されている。この仕組みでは D-Case が扱うことのできる実行時情報が Linux と D-RE を搭載したシステムから得られるそれに限られるため、汎用性が低い。そこで我々は D-Case 側でプログラムを実行させ、その中で SNMP をはじめとする多様な情報源にアクセスして実行時情報を取得することを可能とした。

図 19 に、動的な実行情報に基づいて D-Case に色付けした例を示す。多様な情報源からの動的な実行情報収集を可能とすることで、リアルタイムなディペンダビリティメトリクスの算出がより一般的に可能となり、応用範囲が広がる。

②　研究成果の今後、期待される効果

今後、次のような領域を対象として応用を図っていく。

プラント可視化 ゴミ処理場や化学プラントなどでは多様なセンサ情報が収集され、その値に応じてプラントの状態を判断している。このとき、プラントをシステムととらえそれ全体のディペンダビリティを議論するために D-Case を応用可能である。

ネットワーク管理 LAN や WAN を含めたネットワークのディペンダビリティは、ネットワーク管理者の勘に依るところが大きく、構造的にそれを管理する手法がない。D-Case を用いることで、各種ネットワーク装置の実行状態を含めたディペンダビリティ管理が横断的に可能となる。

§5　研究成果等

(1)　ソフトウエア/ハードウエア

1. ディペンダブル通信機構 (実用化)

概要： トランスポートプロトコル SCTP を拡張して切断耐性の高い通信機構を実現した。FreeBSD7.0 以上および Linux 2.6 に組み込まれ、全世界で実用化されている。

2. ディペンダブル消費電力管理機構 (試作品)

概要： バッテリ駆動 Linux 機器の消費電力を再粒度に管理、制御する機構を実現した。pSurvive システムとして Android MarketPlace で公開している。

3. D-Case およびそれに付随するツール群 (試作品))

概要： ディペンダビリティに関する合意形成ツールとして D-Case を策定し、そのエディタをはじめとするツールを他チームと協力して構築した。東京大学石川チームがホストするウェブサーバにおいて公開している。

4. 実験用マイクロユビキタスノード NUTS

概要： 各種センサを備えた Linux 駆動のマイクロユビキタスノードである。

(2)　規格

(3)　知財出願

①　国内出願 (1 件予定 (D-Case 関連))

②　海外出願 (0 件)

③　その他の知的財産権

(4)　原著論文発表 (国内 (和文) 誌 0 件, 国際 (欧文) 誌 13 件)

1. Michio Honda, Yoshifumi Nishida, Jin Nakazawa, Hideyuki Tokuda, “Performance Enhancement of Transport Layer Handover on Single-homed Mobile Nodes”, IEICE Transactions on Communications, Special Issue on New Challenge for Internet Technology and its Architecture, October. 2007 pp.pp2683-pp.2692

2. Michio Honda, Jin Nakazawa, Yoshifumi Nishida, Masahiro Kozuka and Hideyuki Tokuda, ”A Connectivity-Driven Retransmission Scheme Based On Transport Layer Readdressing”, In Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS2008), pp.277-285, Beijing, China, June 2008.

3. Masato Mori, Junichi Yura, Jin Nakazawa, and Hideyuki Tokuda, ”P-Survive: Process level energy reservation for networked sensing systems,” In Proceedings of the 5th International Conference on Networked Sensing Systems (INSS2008), pp.242-245, Kanazawa, Japan, June 2008.

4. Hiroshi Sakakibara, Takuro Yonezawa, Jin Nakazawa and Hideyuki Tokuda, "Area-Based Activity Information Preservation Mechanism for Ubiquitous Sensor Environment," In Proceedings of the 5th International Conference on Networked Sensing Systems (INSS2008), Kanazawa, Japan, pp119-122, June 2008.

5. Soko Aoki, Jin Nakazawa, and Hideyuki Tokuda, "Spinning Sensors: A Middleware for Robotic Sensor Nodes with Spatiotemporal Models," In Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA2008), pp89-98, Kaohsiung, Taiwan August 2008.

6. Yutaka Ishikawa, Hajime Fujita, Toshiyuki Maeda,Midori Sugaya, Mitsuhisa Sato, Toshihiro Hanawa, Yuki Kinebuchi, Tatsuo Nakajima, Jin Nakazawa, and Hideyuki Tokuda "Towards an Open Dependable Operating System", 12th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009), pp.20-27, 日本, 2009 年 3 月

7. Vu Thi Huong Giang, Honda Michio, Sakakibara Hiroshi, Tokuda Hideyuki, "iPath: Achieving high-performance end-to-end paths for multi-homed mobile hosts,", Third International Conference on Communications and Electronics (ICCE2010), pp.24-29, Vietnam, August 2010.

8. Jin Nakazawa, Yutaka Matsuno, Hideyuki Tokuda, "Evaluating Degree of Systems' Dependability with Semi-Structured Assurance Case", In proceedings of 13th European Workshop on Dependable Computing, pp.111-112, Pisa, Italy, May 2011.

9. Masato Mori, Michio Honda, Jin Nakazawa, Hideyuki Tokuda, "pSurvive: A Process Lifetime Reservation System with Fine-grained Energy Monitoring for Multifunctional Mobile Nodes", IEEE Energy-Aware Solutions Workshop on International Wireless Communication and Mobile Computing Conference(IWCMC), pp.1327-1332, Istanbul, Turk, July 2011.

10. Tetsuro Horikawa, Michio Honda, Jin Nakazawa, Kazunori Takashio, Hideyuki Tokuda, "PACUE: Efficient Heterogeneous Processor Allocations in PCs", Workshop on UnConventional High Performance Computing 2011 (UCHPC 2011) in conjunction with Euro-Par 2011, Bordeaux, France, August 2011.

11. Midori Kato, Michio Honda, Hideyuki Tokuda, "Extending TFWC towards Higher Throughput", IEEE Consumer Communications and Networking Conference (CCNC), 2012, January, Las Vegas, USA

12. Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, Hideyuki Tokuda, "Is it Still Possible to Extend TCP?", ACM Internet Measurement Conference (IMC), 2011, November, Berlin, Germany

13. Jin Nakazawa, Hideyuki Tokuda, "A Three-tier Architecture for User-centric Ubiquitous Networked Sensing", ISRN Journal on Communications and Networking, 2012 (to appear)

(5) その他の著作物 (総説, 書籍など)

1. 徳田英幸、センサネットワーク総論、計測と制御、第４６巻、第２号、2007 年 2 月

2. 中澤 仁, 徳田 英幸, センサアクチュエータネットワークの情報処理基盤 (＜特集＞センシングネットワーク), 情報処理, 情報処理学会, 2010-09-15, 51 巻 9 号, pp1127-1135

3. Hideyuki Tokuda, Jin Nakazawa, “SenseCampus: Sensor enabled Cyber-Physical Coupling for Ubiquitous Services”, 情報処理学会論文誌, 2011 年 12 月

(6) 国際学会発表及び主要な国内学会発表

① 招待講演 (国内会議 6 件, 国際会議 9 件)

1. Yutaka Ishikawa, Hajime Fujita, Toshiyuki Maeda,Midori Sugaya, Mitsuhisa Sato, Toshihiro Hanawa, Yuki Kinebuchi, Tatsuo Nakajima, Jin Nakazawa, and Hideyuki Tokuda ”Towards an Open Dependable Operating System”, 12th IEEE International Symposium on Object, component, service-oriented Real-time distributed Computing (ISORC 2009) 日本，2009 年 3 月

2. 徳田英幸, サイバー・フィジカルカップリングによるユビキタスサービスの実現, WebDB Forum 2011, 2011 年 11 月（国内、基調講演）

3. H. Tokuda, “ Ubiquitous Services: Enhancing a Service with Smart Enabler ”, Symposium on Interaction with Smart Artifacts 2011, 2011 年 3 月 (国際、招待講演)

4. H. Tokuda, “ Challenges in creating smart space everywhere for a smarter society ”, The 16th International Conference on Virtual Systems and Multimedia, 2010 年 1 0 月 (国際、招待講演)

5. 徳田英幸, ユビキタス社会からみた新世代ネットワークの展望, 電子情報通信学会　新世代ネットワーク時限研究専門委員会, 2010 年 10 月（国内、招待講演）

6. H. Tokuda, “ Towards a Ubiquitous Network Society: Challenges in Creating Smart Space Everywhere ”, The 8th International Conference on Pervasive Computing (Pervasive2010) 2010 年 5 月　（国際、基調講演）

7. H.Tokuda,“Cyber-Physical Coupling in Creating Embedded Ubiquitous Services”, 21st Century User-Centric Cyber-Physical Systems Workshop (UCCPS) 2009 年 12 月（国際、招待講演）

8. 徳田英幸, Issues in Urban Sensing and Services 〜 What can you offer after sensing? 〜, IPSJ/IIEICE 情報科学フォーラム (FIT) 2009, 2009 年 9 月 (国内、招待講演)

9. 徳田英幸, “ Challenges in Creating Ubiquitous Services Everywhere 〜From Smart Furniture to Network Robots〜 “ 情報処理学会ユビキタスシステム研究会, 200 9 年 7 月 (国内、招待講演)

10. H. Tokuda,“Security and Privacy Issues in Creating Ubiquitous Services”, The first International Privacy and Security Conference (IPSC2008) 2008 年 1 月（国際、招待講演）

11. H. Tokuda, “ Challenges in Creating Embedded Ubiquitous Services Everywhere ”, IEEE 6th Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia2008), 2008 年 10 月 (国際、基調講演)

12. 徳田英幸, “ ユビキタスネットワークの展望 “ , 電子情報通信学会東京支部シンポジウム, 2008 年 10 月（国内、招待講演）

13. 徳田英幸, これからのユビキタスコンピューティング, IPSJ/IIEICE 情報科学フォーラム (FIT) 2008, 2008 年 9 月 (国内、招待講演)

14. H. Tokuda, ”Challenges in Realizing a Ubiquitous Network Society Development of Smart Spaces and Ubiquitous Services ”, IPSJ Asia-Pacific Software Engineering Conference (APSEC) 2007 (国内、基調講演)

15. H. Tokuda, “Challenges in Ubiquitous Convergence Technology From Smart Space to Networked Robots , International Conference on Ubiquitous Convergence Technology ICUCT 2006 (国際、基調講演)

② 口頭発表 (国内会議 10 件, 国際会議 2 件)

1. 徳田英幸, 高汐一紀, 中澤仁, 岩井将行, 由良淳一 「マイクロユビキタスノード用ディペンダブル OS の実現へ向けて」, 情報処理学会ユビキタスコンピューティングシステム研究会, 2006 年 11 月.

2. Jin Nakazawa, Junichi Yura, Kazunori Takashio, Hideyuki Tokuda: An Operating System Support for Dependability of Micro Ubiquitous Nodes, In Proceedings of the First International Workshop on Dependable Ubiquitous Nodes (IWDUN2007), Tokyo, Nov. 2007.

3. Naoya Namatame, Jin Nakazawa, Kazunori Takashio and Hideyuki Tokuda, ”Life2Guard: A Physical Disorder Detection in Private Rooms,” In Proceedings of the 2nd International Workshop on SensorWebs, Databases and Mining in Networked Sensing Systems (SWDMNSS 2008), pp.177-183, Turku, Finland, July 2008.

4. 青木崇行, 中澤仁, 高汐一紀, 徳田英幸, ”モバイルセンサノードとスマートスペースの協調分担処理機構の実現”, 情報処理学会 第 18 回ユビキタスコンピューティングシステム研究会, pp.17-24, 北海道小樽市, 2008 年 5 月.

5. 菅谷 みどり，藤田肇, 塙敏博, 中澤仁, 松田元彦, 前田俊行, ”ディペンダブルな組込み OS の提案”, 第１０回組込みシステム技術に関するサマーワークショップ (SWEST10) 予稿集,pp.35-38, 2008 年 9 月

6. 中澤仁, 松野裕, 菅谷みどり, 塙敏博, 前田俊行, 藤田肇, 石綿陽一, 杵渕雄樹, 高村博紀, 松田元彦, 三浦信一, 山田浩史, 石川裕, ”オペレーティングシステムおよび実システムにおけるディペンダビリティの評価と見積り”, 日本ソフトウエア科学会第 7 回ディペンダブルシステムワークショップ (DSW’09summer), pp.27-41, 北海道函館市, 2009 年 7 月

7. 加藤真平, 藤田肇, 中澤仁, 松田元彦, 前田俊行, 杵渕雄樹, 塙敏博, 三浦信一, 石綿陽一, 松野裕, 高村博紀, 山田浩史, 吉田哲也, 倉光君郎, 菅谷みどり, 石川裕, ”ディペンダブルシステム向けベンチマークフレームワークの提案”, ソフトウエア科学会第 7 回ディペンダブルシステムワークショップ (DSW’09summer), pp.171-178, 北海道函館市, 2009 年 7 月

8. 森 雅智, 由良 淳一, 中澤 仁, 徳田 英幸, “ モバイル端末におけるアプリケーション動作時間予約機構 ”, 情報処理学会 マルチメディア通信と分散処理研究会 (DICOMO), 岐阜県下呂市下呂温泉, 2010 年 7 月

9. 森 雅智, 由良 淳一, 中澤 仁, 徳田 英幸, “ 高精度資源管理によるモバイルノードのディペンダビリティ向上 ”, 日本ソフトウェア科学会 ディペンダブルシステムワークショップ, 北海道亀田郡七飯町西大沼温泉, 2010 年 7 月

10. 米川賢治, 米澤拓郎, 中澤　仁, 徳田英幸, “ 無線センサネットワークにおける低消費電力な時刻推測 ”, 電子情報通信学会ユビキタスセンサネットワーク研究会, 2010 年 7 月

11. 中澤　仁, 徳田英幸,“ユビキタスセンサネットワークにおけるディペンダビリティ向上のための一考察”, 電子情報通信学会ユビキタスセンサネットワーク研究会, 2011 年 5 月

12. 中澤　仁, 徳田英幸,“D-Case を用いたユビキタス・センサ・ネットワークのモニタリング機構”, 電子情報通信学会ユビキタスセンサネットワーク研究会, 2012 年 1 月

③　ポスター発表 (国内会議 2 件, 国際会議 5 件)

1. Michio Honda, Jin Nakazawa, Yoshifumi Nishida and Hideyuki Tokuda, ”Connectivity-Driven Flow Recovery for Time-Sensitive Transport Services,” In Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN2008), pp.555-556, Montreal, Canada, October 2008.(査読付き)

2. 菅谷 みどり，藤田肇, 塙敏博, 中澤仁, 松田元彦, 前田俊行, ”ディペンダブルな組込み OS の提案”, 第１０回組込みシステム技術に関するサマーワークショップ (SWEST10), 浜松市，2008 年 9 月 4 日．(査読付き)

3. Michio Honda, Hideyuki Tokuda, ”Implementation of SCTP Mobility in The Linux Kernel”,IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2011), Toyama, Japan, August 2011.(査読付き)

4. Masato Mori, Jin Nakazawa, Hideyuki Tokuda, ”Lifetime Reservation for High-priority Applications on Multifunctional Mobile Node”, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Toyama, Japan, August 2011.(査読付き)

5. Midori Kato, Michio Honda, Hideyuki Tokuda, ”Implementation of Congestion Control for Multimedia Streaming”, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Toyama, Japan, August 2011. (査読付き)

6. 米川賢治, 米澤拓郎, 中澤仁, 徳田英幸, ”無線センサネットワークにおけるモバイルシンクノードのためのデータ収集プロトコル”, ソフトウェア科学会第 28 回大会, 那覇市, 2011 年 9 月

7. Midori Kato, Michio Honda and Hideyuki Tokuda, ”A practical congestion control algorithm for real-time multimedia streaming”, Grace Hopper Conference (GHC), Nov 2011, Portland, U.S.A.

(7)　受賞・報道等

①　受賞

1. 情報処理学会功績賞　徳田英幸、2011 年 6 月

2. Applied Networking Research Prize by Internet Research Task Force (http://irtf.org/anrp), Taipei, 82nd IETF meeting, 2011 年 11 月

3. Featured on Slashdot -“Middleboxes vs. the Internet’s End-to-End Principle”, 2011 年 8 月

4. 電子情報通信学会 USN 研究賞 -“ユビキタスセンサネットワークにおけるディペンダビリティ向上のための一考察”, 2012 年 1 月

② マスコミ (新聞・ＴＶ等) 報道

③ その他

1. SFC Open Research Forum 2006、2006 年 11 月、六本木ヒルズ
2. SFC Open Research Forum 2007、2007 年 11 月、六本木ヒルズ
3. SFC Open Research Forum 2008、2008 年 11 月、六本木ヒルズ
4. SFC Open Research Forum 2009、2009 年 11 月、六本木ヒルズ
5. Embedded Technology 2009, 2009 年 12 月、パシフィコ横浜
6. SFC Open Research Forum 2010、2010 年 11 月、六本木ヒルズ
7. Embedded Technology 2010, 2010 年 12 月、パシフィコ横浜
8. SFC Open Research Forum 2011、2011 年 11 月、東京ミッドタウン
9. Embedded Technology 2011, 2011 年 12 月、パシフィコ横浜

(8) 成果展開事例

① 実用化に向けての展開

1. ディペンダブル通信機構 (実用化)

概要： トランスポートプロトコル SCTP を拡張して切断耐性の高い通信機構を実現した。FreeBSD7.0 以上および Linux 2.6 に組み込まれ、全世界で実用化されている。

② 社会還元的な展開活動

§6　研究期間中の主な活動 (ワークショップ・シンポジウム等)

年月日	名称	場所	参加人数	概要
2007.11.25	The First International Workshop on Dependable Ubiquitous Nodes (IWDUN2007)	東京電機大学	15	ディペンダブルユビキタスコンピューティングとそれを支えるシステムに関する国際ワークショップ
2008.7.31	The Asian Europian Workshop on Ubiquitous Computing (AEWUC)	Oulu University, Oulu, Finland	30	ユビキタスコンピューティングと組込システムに関する国際ワークショップ
2009.5.11-14	Seventh International Conference on Pervasive Computing (Pervasive 2009)	奈良県公会堂	300	パーベイシブコンピューティングと組込システムに関する国際会議
2009.6.17-19	Sixth International Conference on Networked Sensing Systems (INSS2009)	Carnegie Mellon University, Pittsburgh, USA	300	ネットワークドセンシングシステムとと組込システムに関する国際会議
2009.8.3-4	The Asian Europian Workshop on Ubiquitous Computing (AEWUC)	ラフォーレ修善寺	40	ユビキタスコンピューティングと組込システムに関する国際ワークショップ

年月日	名称	場所	参加人数	概要
2010.5.15	The Asian Europian Workshop on Ubiquitous Computing (AEWUC)	Oulu University, Oulu, Finland	30	ユビキタスコンピューティングと組込システムに関する国際ワークショップ
2011.3.7-9	Symposium on Interaction with Smart Artifacts	東京大学駒場キャンパス	50	組込センシングシステムとその応用に関する国際ワークショップ
2011.8.28-31	The 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2011)	富山国際会議場	150	実時間システムと組込システムとその応用に関する国際会議

§7 結び

本研究では、情報家電機器等で利用されているミッドレンジのハードウェアプラットフォームだけではなく、さらに小型軽量で、エネルギー効率の良い電池駆動可能なマイクロレンジのハードウェアプラットフォーム (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した。(1) 複数種類のネットワークインタフェースを持つ次世代オープン端末のためのディペンダブル通信機能（研究課題 1)、ならびに (2) バッテリ電力の詳細なロギングと予約に基づくディペンダブル消費電力管理機能（研究課題 2）に加え、(3)D-Case およびディペンダビリティメトリクスを研究課題に定めて実施した。顕著な成果として、(1) ディペンダブル通信機構 (実用化)、(2) ディペンダブル消費電力管理機構 (試作品)、および (3)D-Case およびそれに付随するツール群 (試作品) を得ており、特に (1) は FreeBSD や Linux といった広範囲に使用されているオペレーティングシステムに組み込まれて実用化した。また、(3) では D-Case サブコアチームメンバと連携して D-Case エディタや D-Case ビューワを含む各種のツール開発を実施した。さらに、研究機関を通して質の高い国際会議に研究成果を発表した。以上より本研究では、要素技術の開発にとどまらず、プロジェクト全体に関するソフトウエアエンジニアリングの分野についても貢献した。

参考文献

[1] R. Stewart. Stream Control Transmission Protocol. *RFC 4960*, Sep. 2007.

[2] V. Paxson and M. Allman. Computing TCP's Retransmission Timer. *RFC 2988*, Nov. 2000.

[3] SCTP BSD kernel implementation. http://www.sctp.org/.

[4] Alex C. Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 155–166, New York, NY, USA, 2000. ACM Press.

[5] D. Funato, K. Yasuda, and H. Tokuda. TCP-R: TCP mobility support for continuous operation. *ICNP*, Vol. 00, p. 229, 1997.

[6] S. J. Koh, H. Y. Jung, J. H. Min. Transport Layer Internet Mobility based on mSCTP. *IEEE ICACT*, Vol. 1, pp. 329–333, 2004.

[7] R. Moskowitz, P. Nikander. Host Identity Protocol (HIP) Architecture. *RFC 4423*, May. 2006.

[8] D. Johnson, C. Perkins, J. Arkko. Mobility Support in IPv6. *RFC 3775*, Jun. 2004.

[9] F. Teraoka, Y. Yokote and M. Tokoro. A Network Architecture Providing Host Migration Transparency. *ACM SIGCOMM*, pp. 209–220, Sep. 1991.

[10] Ramón Cáceres and Liviu Iftode. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. *Selected Areas in Communications, IEEE Journal on*, Vol. 13, No. 5, pp. 850–857, Jun. 1995.

[11] Simon Schütz, Lars Eggert, Stefan Schmid, and Marcus Brunner. Protocol enhancements for intermittently connected hosts. *SIGCOMM Comput. Commun. Rev.*, Vol. 35, No. 3, pp. 5–18, 2005.

[12] Moonjeong Chang, Meejeong Lee and Seokjoo Koh. Transport Layer Mobility Support Utilizing Link Signal Strength Information. *IEICE Transactions on Communications*, Vol. 87, pp. 2548–2556, 2004.

[13] D. Miyakawa and Y. Ishikawa. Process oriented power management. *SIES '07.*, pp. 1–8, 2007.

[14] Enhanced Intel SpeedStep Technology How To Document. http://www.intel.com/cd/channel/reseller/asmo-na/eng/203838.htm.

[15] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pp. 1–11, 2003.

[16] X. Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 186–195. ACM Press New York, NY, USA, 2007.

[17] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pp. 28–32. ACM Press New York, NY, USA, 2007.

[18] Tim Kelly and Rob Weaver. The goal structuring notation - a safety argument notation. In *Proc. of the Dependable Systems and Networks 2004, Workshop on Assurance Cases*, 2004.

[19] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, Vol. 1, No. 1, pp. 11–33, Jan.-March 2004.

[20] Keishi Okamoto and Makoto Takeyama. An approach to assurance cases. In *proceedings of Dependable System Workshop (DSW'09)*, 2009.