レジリエントシステムの理論と設計
ーオポチュニスティック型資源活用システムー

代表研究者	笠松　大佑	パデュー大学	計算機科学部・博士研究員
共同研究者	Leszek T. Lilien	西ミシガン大学	計算機科学部・准教授
		パデュー大学	計算機科学部・非常勤准教授

1 概要

現代文明は、ますます複雑化する巨大なシステムに支えられている。システム脅威（特に大規模災害）の進展は、これらのシステムに新たな問題を提起している。本研究では、Failure-accepting システムにおける、レジリエンスのシステム設計コンセプトへの新しいアプローチを提案する。本システムでは、オープンかつ協調な環境において、予期しないシステム脅威に対処可能であることを目指す。本報告書では、以下の三点について記述する。第一に、緊急事態救助・復旧活動のための、オポチュニスティック型資源活用システム（Oppsys）を用いたレジリエントシステムのコンセプトを提案した。Oppsys の最大の特徴は、システムの事前配備計画において、システム全体を完全に事前設計する必要がなく、むしろそれがほとんど要求されないことにある。代わりに、Oppsys はアドホックにそのシステムリソースを拡張かつ縮退する。第二に、システム復旧可能性問題として、確率を用いたランダムグラフ理論に基づいて、サービスの許容可能なレベルへのシステム復旧について、問題の定式化を行った。本問題を解決するために、Oppsys によるリソースの割当、再配置、解放を制御するためのリソース制御アルゴリズムを開発した。簡単なグラフの例を用いて、本アルゴリズムの基本的な動作を確認した。第三に、システムの試験ツールとして、Oppsys を構成する各システム要素の基本機能を仮想化する Oppsys 仮想マシンを設計した。

2 研究目的

2-1 レジリエントシステム

現代文明は、技術の進展に伴い、ますます複雑化する巨大な情報システムに支えられており、システムの故障やシステムへの攻撃に対する事例がますます増加している。万が一、そのようなシステムが故障すれば、現代社会への損害は深刻かつ計り知れないものとなる。多くの複雑システムは、システム運用に影響を与えるたくさんの脅威に直面している。

本研究では、システムへの脅威の一つとして、広範囲なエリアに渡るシステムの故障を引き起こすことを特徴とする大規模災害に着目する。そこでは、ハードウェアの破壊により、正常なシステムオペレーションを妨げ、意思決定者による情報へのアクセスを制限し、復旧の選択肢を限定する。代表的な事例を以下に示す。２００５年８月２９日、ハリケーン・カトリーナはアメリカの東南地方（ルイジアナ州とミシシッピ州）に大規模な損害をもたらした。特に、１３４の大規模ネットワークにおいて通信断絶となった[2]。ほとんどの損害は停電によるものであり、それらのほとんどは復旧に１０日を要した。また、２０１１年３月１１日、世界史上４番目に強力なマグニチュード９の東日本大震災が発生した。地震によって引き起こされた約１５メートルの津波により、電力源を失い、福島第一原発事故が発生した。地震と津波は、日常生活、食糧の生産と供給、商業活動、交通機関に深刻な損害を与えた。通信システムにおいては、３８５の固定電話の基地局と４９００の携帯電話の基地局が崩壊した[3]。影響を受けた設備の約１０％は、震災より二週間後の３月末でも、いまだ運用停止であった[3]。

研究、開発、工学において、複雑システムへの新しい問題に対処することが極めて重要であることは、広く認知されている。従来のシステムコンセプト（reliability, fault tolerance, survivability, security, dependability or robustness）は、failure-avoiding システム、isolated システム、予期可能なシステム脅威のいずれかに焦点がある。新しい問題に対処するために、オープンかつ協調な環境において、予期しないシステム脅威に対処可能である Failure-accepting システムが求められる（図１）。予期しない脅威とは、予期しない故障、攻撃、事故であり、特に大規模災害（地震、津波、洪水、ハリケーン、氷嵐、広範囲の停

電など）である。オープンシステムとは、未知のシステムと意図せずに相互作用し、参加・離脱する可能性のある情報システムの集合から成る。故障許容（Failure acceptance）とは、情報システムが復旧可能な範囲内での故障を許容できる性質である。本研究では、予期しないシステム脅威に対処するためのシステムの能力を強調し、そのようなシステムをレジリエントシステムと呼ぶ。システムのレジリエンスは以下のように定義されている：（１）変化に直面する際に、正当に信頼されるシステムによるサービス提供の存続[4]、（２）サービスの通常運用への様々な故障や問題に直面する際、サービスの許容可能なレベルを提供・維持するためのシステムの能力[1]。変化とは、コンピューティング・システムの進展やシステム脅威の進展である[4]。また、問題とは、システムの通常運用に影響を与えるシステムの特徴、脅威、脆弱性、欠陥、エラー、故障、攻撃、事故などである[1]。

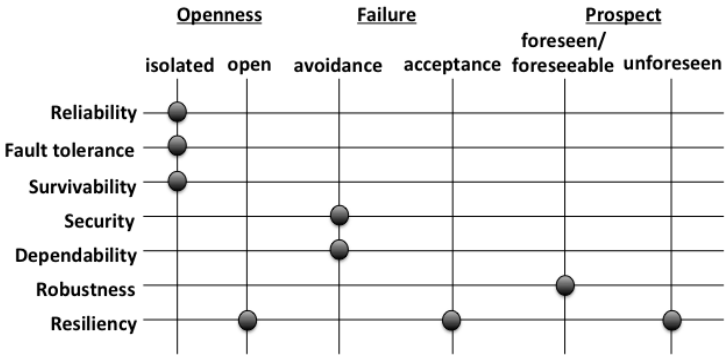


図１　システム設計コンセプトの比較

2-2 従来システム対 Oppsys における予期しない脅威に対するレジリエンスの実現方法

従来システム（P2P システム[5]、グリッド・システム[6]、クラウド・システム[7]、アドホックネットワーク・システム[8]）において、各システム要素は、システムのサイズや事前に設計されたコンポーネントの位置に従って全て一緒に展開される。そのようなシステムにおけるレジリエンスの提供は、各個別のシステム内に冗長性の設計を要求する。多くのそのような従来システムが開発されている。

本研究では、オポチュニスティック型資源活用システム（Oppsys: Opportunistic resource utilization systems）[10]を用いたレンジリエンスの確保に着目する。Oppsys の最大の特徴は、システム配備前に完全に事前設計する必要がなく、代わりに、アドホックにシステムを拡張かつ縮退できることにある。Oppsys は、システムの環境における利用可能なリソースを想定し、日和見的に所望のリソースを獲得することで拡大する。このアドホックなシステムの拡張・縮退により、システムの事前配備計画（最終的なシステムサイズ、コンポーネントの位置など）は極めて小さくなる。したがって、従来システムとは対称的に、Oppsys におけるレジリエンスは、各個別のシステム内に冗長性を設けることなしに提供できる。このため、ダイナミックかつ予期しない状況（緊急事態救助・復旧活動など）において、Oppsys は高信頼な運用を達成するために適用可能である。

Oppsys ノードは以下のノードで構成される（図２）：（１）Seed Oppsys：ネットワークの初期配備の段階で事前に設計され、一緒に展開されるノードの集合、（２）分散制御ノード：Seed ノードの部分集合、（３）通常の Helper：他のネットワークにおける外部ノード（例えば、コンピュータ・ネットワーク、携帯電話通信インフラ、衛生、ホーム・ネットワーク、WiMAX ネットワーク、乗物用通信システムなど)、（４）制限付 Helper（例えば、Oppsys 可能なセンサーノードなど)。

Oppsys または Oppsys を含むシステムに対し、基本的なシステム復旧オペレーションを以下に示す：（１）故障したシステム内の Seed Oppsys は復旧オペレーションを始め、システム復旧のために拡張する必要があるか決定する、（２）Seed Oppsys はシステムの環境における潜在的な Helper を発見し、Helper 候補を特定する、（３）Oppsys の制御ノードは、十分な Helper 候補か評価する、（４）十分な Helper が特定される場合、制御ノードは最もよい Helper を選択し、Oppsys 内にそれを承認する、（５）Oppsys は承認した Helper のリソースを統合する。以上より、元のシステムの存続した要素（Seed Oppsys を含む）と新たに統合した Helper によって、復旧したシステムはサービスの許容可能なレベルを提供する。

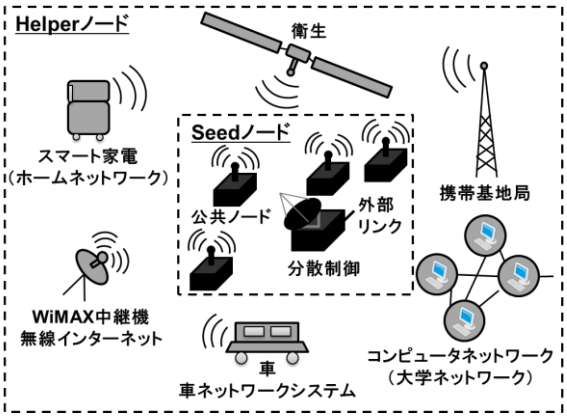


図２　Oppsys

3 研究方法

Oppsys を用いて、レジリエンスの確保を試みる際、いくつかの理論と設計上の問題がある。

問題１：レジリエンスのためのリソースの制御方法。システムに冗長なリソースを提供する代わりに、Oppsys はシステムの運用中の環境において利用可能なリソースを活用することでレジリエンスを実現する。しかしながら、本アプローチは、そのようなリソースを制御するための手法と、そのような制御を行うための時間を必要とする。しかしながら、ほとんどの緊急時は、制御の決定をなすために十分な時間を残さない。人手による意思決定手法は多くの時間を要することから、システム復旧オペレーションの自動化が要求される。

問題２：レジリエンスのためのリソースの計測方法。DeMarco は、「計測できないことは制御できない」と述べた[9]。レジリエンスの概念上の定義は提供されているが[1, 4]、定量化がなされていない。従来システムの設計コンセプトにおいて、コンポーネントレベルの評価技術（信頼性、可用性、フォールト・トレランスなど）がたいてい用いられる。システムがより複雑かつ予期できないものになるのに伴い（トポロジー、位置、コンポーネント、振る舞いなど)、評価技術において考慮すべき変数の数は大幅に増加する。結果として、従来の評価技術はオフライン（事前配備の段階）で使用され、オンラインでの使用に十分に有効ではない[4, 15]。例えば、Laprie は、「従来、検証と評価はオフライン（事前配備）で行われる。そのようなアプローチは進展するシステムの場合、明らかに不足である。評価は実行時（運用中）に行われなければならない。」と述べた[4]。同様に、Al-Kuwaiti は、「システムがより複雑かつ境界のないものになるのに伴い、変数の数は大幅に増加し、コンポーネントレベルの観点から開発された定量的なモデルの使用は非実用的なものとなった。」と述べた [15]。したがって、従来の評価技術は進展するダイナミックなシステムのモデル化と制御に対し、十分ではない。

本研究では、以下の研究に取り組む：（１）Oppsys 構成理論の構築：Oppsys によるシステム復旧を実現するためのシステム構成理論を構築する。Helper ノード候補の評価尺度の数学的な定義と、統合する Helper ノードの選択アルゴリズムの設計を行う。（２）Oppsys 試験ツールの開発：アプリケーションベースの特性分析、性能評価を行うための試験環境として、エミュレータを開発する。

4 研究結果

4-1 システム復旧可能性問題のモデル化とリソース制御アルゴリズムの開発

（１）表記

v_i	点 i（vertex）
e_i	枝 i（edge）
$(v_i,\ v_j)$	点 v_i と v_j を接続する無向枝
$P(x)$	変数 x の確率
$G(V,\ E)$	点集合 V と枝集合 E から成るランダムグラフ G、そこでは確率指標の変数が各点と枝に付加される[11]
G'	G の点集合と枝集合の部分集合から成る部分グラフ

$\|M\|$	集合 M の要素数
K	V における２つ以上の特定の部分集合、すなわち $K \subseteq V,\ 2 \leq \|K\| \leq \|V\|$
T_i	最小パス集合 i
St	構造関数、すなわち全ての最小パス集合の総和

（２）定義

グラフにおけるパスとは、与えられた２点間を接続する点と枝の集合である[16]。最小パスとは、同一の２点を接続する部分グラフを含まないパスである(あるコンポーネントの除去はパスとならなくなる)[16]。グラフの連結性とは、グラフにおける任意の２点間を接続するパスがあるグラフである[16]。K 点(K>1)とは、点集合における部分集合である[12]。K ツリーとは、任意の K 点に対する連結性を確保する最小パスを持つグラフである[12]。

（３）準備：K ツリー生成アルゴリズムと包除原理

アルゴリズム１は、グラフにおける全ての K ツリーを列挙するための *GenK-tree* アルゴリズム[12]を示す。ステップ１は、K ツリーとなる V_S を作成する：(１)任意の点 v と、(２)K ツリーに既に含まれた枝 $e=(u, z)$ の点 u の２つを用いる。v とは現在探索中の点であり、u と z は各々K ツリーに既に含まれた枝の集合に対する始点と終点である。ステップ２は、新しい K ツリー（V_S と E_S）を出力する。ステップ３は、K 点を含まない任意の接続されるコンポーネントを除去する。ステップ４は、既に探索された枝集合を空集合にする。ステップ５と６は、別の K ツリーを探索するために、再帰処理により K ツリーを探索し続ける。

Algorithm 1. GenK-tree [12] using C-style pseudo code.

Input: G, E_S, H, v — where G is a graph; E_S is a set of edges that have been included in the K-Tree (E_S will become the set of edges of a K-tree); H is a set of edges already found, used to prevent enumerating duplicate K-trees; and v is a the vertex being currently searched.

Initial Input: (G, ø, ø, v [1]), where v [1] is one of the K-vertices, and ø denotes an empty set

Output: All K-Trees in G

Algorithm Steps:

```
void function GenK-tree (G, E_S, H, v)
1.    V_S = {u | there exists z, (u, z) ∈ E_S};
      /* V_S will become the set of vertices of a K-tree */
      V_S = V_S ∪ {v};   // v is the last vertex added to the K-Tree
2.    if ( K ∈ V_S ) { print V_S and E_S; return; }
3.    Compute the set of connected components after removed V_S from V in G and removed all edges
      connecting to V_S from E in G. Let the components be G_1, G_2, ..., G_Q, where the set of components
      includes at least one of the K-vertices. Ignore the other components.
4.    H_0 = ø;
5.    if ( v ∉ K ) {
          for ( each edge e = (v, v_y) ∉ H_0 )
          /* v_y belongs to one of G_1, G_2, ..., or G_Q */
              GenK-tree (G, E_S ∪ {e}, H ∪ H_0, v_y);
              H_0 = H_0 ∪ {e};
          }
      }
6.    else {
          for ( each edge e = (v_x, v_y) ∉ H_0 ) {
          /* v_x ∈ V_S, and v_y belongs to one of G_1, G_2, ..., or G_Q */
              GenK-tree (G, E_S ∪ {e}, H ∪ H_0, v_y);
              H_0 = H_0 ∪ {e};
          }
      }
```

以下の式１は、アルゴリズム１によって列挙されたKツリーから構造関数の確率を計算するための包除原理を示す[13]。$T=\{T_1, T_2, \ldots, T_t\}$を列挙されたKツリーとし、tは集合$|T|$の要素数である。構造関数の確率は、以下のように表現される：

$$\Pr(T_1 \cap T_2 \cap \cdots \cap T_t) = \sum_i \Pr(T_i) - \sum_{i<j} \Pr(T_i \cap T_j) + \sum_{i<j<k} \Pr(T_i \cap T_j \cap T_k) + \cdots + (-1)^{t+1} \Pr(T_1 \cap T_2 \cap \cdots \cap T_t) \quad (1)$$

（４）数学による問題の定式化

システムツールによるシステム復旧評価の提案は、システム運用者にとって役立つ。また、復旧オペレーションの自動化は、ほとんどの緊急事態が意思決定のために十分な時間を残さないことから重要である。

提案のシステム復旧手法は、Oppsys運用を可能にするサービスの許容可能なレベルへのシステム復旧を確保する。システム復旧のためのリソースの日和見的な使用は、確率の問題である。したがって、レジリエンス計測に対する本アプローチは、確率を用いたランダムグラフモデルを用いて行う。Oppsysを表すランダムグラフGにおいて、点はSeedノードとHelperノードを表し、枝はそれらの有線あるいは無線のリンクを表す。故障したシステム・コンポーネントを表す点と枝は、グラフから除去される。存続したシステム・コンポーネントは点Kとする。

以下の式２と式３は、i番目の点v_iの復旧可能性$RP(v_i)$に対するメトリックを定義する。

$$RP(v_i) = A*\text{Capability} + B*\text{Energy} + C*\text{Availability} \quad (2)$$

$$A + B + C = 1 \quad (3)$$

A、B、Cは３つのメトリック（能力、電力、利用可能性）に対する重みである。能力は、i番目のノードのコンピューティング能力（例えばOppsys可能なセンシング能力）や利用可能な計算資源（例えばCPUやメモリ）によって決定する。故障していない存続したノードの能力は 100%であり、故障したノードのそれは 0%である。電力は、i番目のノードの電力量によって決定する。電池駆動による全ての動作するSeedノードとHelperノードにおける最大の電池残量を100%と表す(異なる電池容量をもつ異なるノード間の電力量の比較を可能とするために正規化する)。また、不十分な電力を持つノードにおける電力量は0%と正規化して表す。電力網に接続したSeedノードとHelperノードにおける一定の電力量は100%である。

利用可能性は、i番目のノードのOppsys準備（訓練）のレベルに関して決定する。よく訓練されたHelperは無訓練のHelperより高い確率である。これは、セキュリティの観点から悪意のあるHelperを避けるための有効な手段となる可能性がある。故障していない存続したノードの利用可能性は100%、故障したノードのそれは0%である。

式４は、i番目の通信リンクを表す枝e_iの復旧可能性$RP(e_i)$に対するメトリックを定義する。

$$RP(e_i) = \text{Link communication metric} \quad (4)$$

通信リンク・メトリックは、i 番目の通信性能（通信速度、電波信号強度、接続時間、パケットロス、遅延）によって決定する。その値は、必要十分なリンクに対する100%から故障したリンクに対する0%の範囲である。

システム復旧可能性$R_K(G)$は、システム復旧の成功確率である。ランダムグラフGにおいて、全てのK点が１つ以上のKツリー（全ての要素は0%より大きい復旧可能性を持つ）によって接続される場合、システム復旧可能性は計算可能である。Lをサービスの許容可能なレベルとする。ここで、システム復旧問題を「点と枝の与えられた復旧可能性を持つ与えられたランダムグラフGについて、$R_K(G)$が閾値Lより大きいという条件の下、部分グラフGを探索せよ」とし、以下のように定式化する：

System recoverability problem:

$$\text{Find } G' \quad (5)$$

Subject to:

$$R_K(G) \geq L \quad (6)$$

システム復旧可能性$R_K(G)$が閾値Lより大きい場合にのみ、システムはサービスの許容可能なレベルで復旧可能である。これは、全てのK点のペアが接続される場合のみである。

Algorithm 2. GenSubgraph using C-style pseudo code.

Input: G, L, A, M—where G is a graph with recoverable potential, $RP(v_i)$, of the vertex v_i (representing the i's system node) and recoverable potential, $RP(e_i)$, of the edge e_i (representing the i's system link); and L is the threshold; A is the type of algorithms and M is the metrics considered in this algorithm

Output: G'—where G' is a subgraph satisfying Equation 6

Algorithm Steps:

```
void function GenSubgraph (G, L, A, M)
1.      G' = ø; v_m = ø; e_m = ø;    // ø denotes an empty set
2.      if ( A is resource assignment/replacement algorithm
            and M is RP (v_i) ) {
            while ( the number of V' is not greater than the
                        number of V ) {
            /* V'/V is a vertex set of a subgraph/graph */
                for ( each vertex v_i ∉ G' )
                { v_m = {v_i | max RP (v_i)}; }
                /* v_m is a vertex with max RP (v_i) in V */
                G' = G' ∪ {v_m};
                GenK-tree (G', ø, ø, v_i); // Algorithm 1; v_i ∈ K
                if ( the number of K-trees isn't smaller than one ) {
                   Computes R_K (G');
                   /* by using the inclusion-exclusion principle */
                   if ( R_K (G') is not smaller than the threshold L )
                  { print G'; return; }
                   /* R_k (G') is recoverability of a subgraph */
                }
            }
        }
3.      else if ( A is resource assignment/replacement algorithm
                    and M is RP (e_i) ) {
            while (the number of E' is not greater than
                       the number of E ) {
            /* E'/E is a edge set of subgraph/graph */
                for ( each edge e_i ∉ G' )
               { e_m = {e_i | max RP (e_i)}; }
                /* e_m is a edge with max RP (e_i) in E */
                G' = G' ∪ {e_m};
                GenK-tree (G', ø, ø, v_i);
                if ( the number of K-trees isn't smaller than one ) {
                    Computes R_K (G');
                    if ( R_K (G') is not smaller than the threshold L )
                   { print G'; return; }
                }
            }
        }
4.      else { print G'; return; }
        /* A is resource relief algorithm */
```

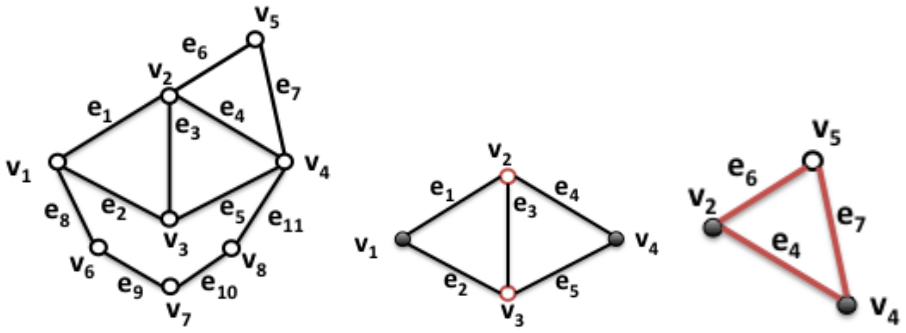
（５）部分グラフ生成アルゴリズム

アルゴリズム２はリソースの割当、再配置、解放を制御するためのアルゴリズム（RARR: resource assignment/replacement/relief) GenSubgraph を示す。本アルゴリズムは、Helper 候補から Oppsys へのリソースを選択することによって、サービスの許容可能なレベルでシステムを復旧するために用いる。アルゴ

リズムにおいて、Oppsys（存続したノード、Seed ノード、Helper ノードを含む）は G と表す。Helper ノード候補から Oppsys によって活用されるリソースの集合は G'と表す。ステップ１は、部分グラフと最大の復旧可能性をもつ点と枝の初期化である。ステップ２は、アルゴリズムのタイプがリソースの割当または再配置、かつアルゴリズムにおいて考慮されるメトリックが点の復旧可能性の場合である。本ステップは最大の復旧可能性を持つ点を選択して、部分グラフに追加する。次に、アルゴリズム１を用いて、G'における全ての K ツリーを列挙する。K ツリーの数が１以上の場合、列挙された全ての K ツリーに対し、式１を用いて部分グラフのシステム復旧可能性を計算する。また、その復旧可能性が閾値 L 以上の場合、G'を出力し、本アルゴリズムを終了する。V'の要素数が V の要素数以下の場合、本アルゴリズムは本ステップを繰り返す。式６を満足する部分グラフを作成できない場合には、何も出力しない。ステップ２では、式２における重み（A、B、C）を調整することで、サービスの許容可能なレベルを変更可能である。ステップ３は、アルゴリズムのタイプがリソースの割当または再配置、かつアルゴリズムにおいて考慮されるメトリックが枝の復旧可能性の場合である。本ステップは、ステップ２と同様の方法で処理される。ステップ４は、アルゴリズムのタイプがリソースの解放の場合である。本ステップは、空集合である G'を出力し、本アルゴリズムを終了する。

（６）数値例

図３は、与えられたサンプルグラフと RARR アルゴリズムによって得られる部分グラフを示す。図３（１）は、与えられたサンプルグラフであり、本アルゴリズムにおける G である。表１は、復旧可能性を示し、本アルゴリズムにおける $RP(v_i)$ と $RP(e_i)$ である。



(1) Sample given graph　(2) Bridge graph ($|K| = 2$)　(3) Triangle graph ($|K| = 2$)

open/filled circle: normal vertices/K-vertices

図３　与えられたサンプルグラフと得られるサブグラフ

表１　点と枝の復旧可能性

	Capability	Energy	Availability		Link communication
v_1	1.0	1.0	1.0	e_1	0.1
v_2	0.9	0.9	0.1	e_2	0.1
v_3	0.9	0.9	0.1	e_3	0.1
v_4	1.0	1.0	1.0	e_4	0.8
v_5	0.1	0.1	0.1	e_5	0.1
v_6	0.1	0.1	0.9	e_6	0.8
v_7	0.1	0.1	0.9	e_7	0.8
v_8	0.1	0.1	0.9	e_8	0.1
				e_9	0.1
				e_{10}	0.1
				e_{11}	0.1

アルゴリズムのタイプがリソースの割当、かつアルゴリズムにおいて考慮されるメトリックが点の能力と電力の場合、本アルゴリズムはステップ２を実行する。例えば、K を v_1 と v_4、L を 0.9、式３における A、B、C を 0.4、0.4、0.2 とする。本ステップは、最大の復旧可能性を持つ点 v_2 を選択し、G'に加える。ここで、$RP(v_2)$ は 0.74 である。次に、アルゴリズム１を用いて、G'における全ての K ツリーを列挙する。K ツリーの

数が１以上の場合、復旧可能性を計算する：

K と v_2 を含むパスに対するシステム復旧可能性：
$St = T_1$, where $T_1 = v_2$
$R_K(G) = Pr(T_1) = 0.74$

これは L より小さいため、本ステップは次のノードとして、v_3 を選択し、G' に追加する。ここで、$RP(v_3)$ は 0.74 である。次に、G' における全ての K ツリーを列挙し、K ツリーの数が１以上の場合、復旧可能性を計算する：

ブリッジグラフ（*bridge graph*）に対するシステム復旧可能性：
$St = T_1 \cup T_2 \cup T_3 \cup T_4$, where $T_1 = v_2$, $T_2 = v_2v_3$, $T_3 = v_3v_2$, $T_4 = v_3$
$R_K(G) = Pr(T_1 \cup T_2 \cup T_3)$
$= T_1 + T_2 + T_3 + T_4 - (T_1 \cap T_2 + T_1 \cap T_3 + T_1 \cap T_4 + T_2 \cap T_3 + T_2 \cap T_4 + T_3 \cap T_4) + (T_1 \cap T_2 \cap T_3 + T_1 \cap T_2 \cap T_4 + T_1 \cap T_3 \cap T_4 + T_2 \cap T_3 \cap T_4) - T_1 \cap T_2 \cap T_3 \cap T_4$
$= v_2 + v_2v_3 + v_3v_2 + v_3 - (v_2v_3 + v_2v_3 + v_2v_3 + v_2v_3 + v_2v_3 + v_2v_3) + (v_2v_3 + v_2v_3 + v_2v_3 + v_2v_3) - v_2v_3$
$= 0.9324$

システム復旧可能性が閾値 L 以上であるため、図３（２）に示すような G' を出力し、本アルゴリズムを終了する。

アルゴリズムのタイプがリソースの再配置、かつアルゴリズムにおいて考慮されるメトリックが枝の通信リンクである場合、本アルゴリズムはステップ３を実行する。例えば、K を v_2 と v_4、L を 0.85 とする。ステップ２と同様の方法で、本アルゴリズムは、図３（３）に示すようなトライアングルグラフ（*triangle graph*）を出力し、そのシステム復旧可能性は 0.896 である。

4-2 Oppsys 仮想マシンの設計

システムの試験ツールとして、Oppsys を構成する各システム要素の基本機能を仮想化する「Oppsys 仮想マシン」を設計した。緊急時のアプリケーションシナリオ例として、センサシステムが整備された町における震災を想定し（図４）、ハードウェアと、Seed ノードや Helper ノードの擬似機能、エミュレーション環境の視覚化機能を持つソフトウェアで構成する（図５）。

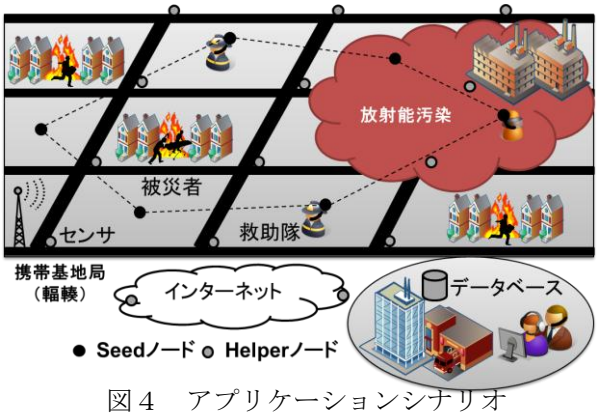


図４　アプリケーションシナリオ

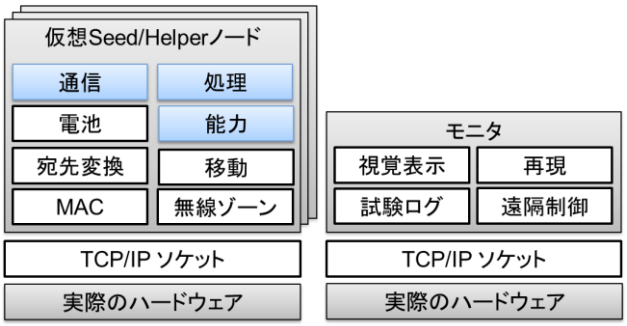


図５　Oppsys 仮想マシン

図６は、アプリケーションシナリオ例に対するシーケンス図を示す。Seed Oppsys が展開された後、Oppsys は通常の Helper（Helper）と制限付 Helper（Lite）を承認することで拡大する。Seed Oppsys は、Helper ノード経由で Lite ノードから情報を収集する。大規模災害において、一つの優先事項は家屋に取り残された生存者を探すことである。例えば、Lite ノードは、Bluetooth 機能を備えたモーションセンサなどである。表２は、シーケンス図における各メッセージに対するプリミティブを示す。各プリミティブに対して擬似コードの開発を行ったが、スペースの都合上、説明を省略する。

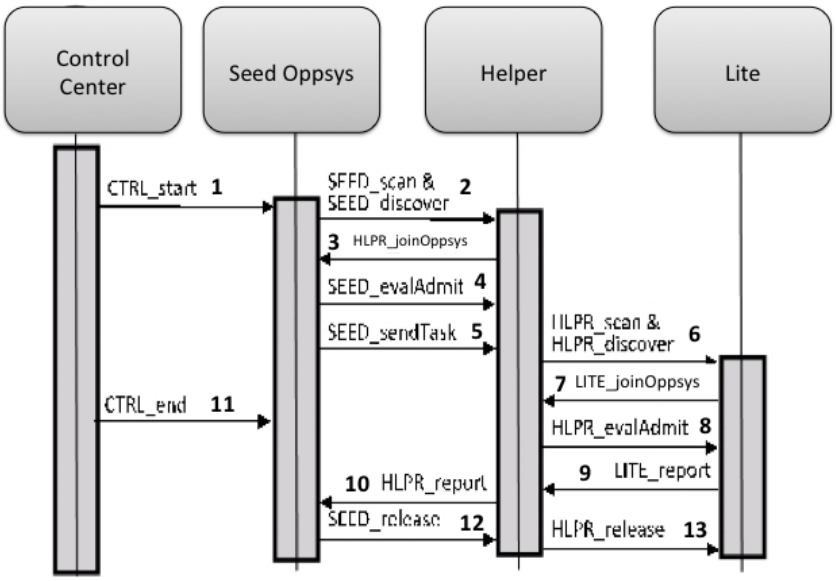


図６　アプリケーションシナリオ例に対するシーケンス図

表２　Oppsys プリミティブ

1	CTRL_start	Initiate oppsys
11	CTRL_end	Terminate oppsys
2	SEED_scan	Scan communication spectrum to detect devices that could become candidate helpers
2	SEED_discover	Discover candidate helpers with a specific communication mechanism
4	SEED_evalAdmit	Evaluate a device and admit it into the oppsys if the device meets criteria for admittance
5	SEED_sendTask	Send a task to another oppnet device
12	SEED_release	Release a helper when no longer needed
3	HLPR_joinOppsys	Tell joining oppsys
6	HLPR_scan	Scan communication spectrum to detect devices that could become candidate helpers
6	HLPR_discover	Discover candidate helpers with a specific communication mechanism
8	HLPR_evalAdmit	Evaluate a device and admit it into the oppsys if the device meets criteria for admittance
10	HLPR_report	Report information to the control center/coordinator
13	HLPR_release	Release a lite when no longer needed
7	LITE_joinOppsys	Tell joining oppsys
9	LITE_report	Report information to the control center/coordinator

５ まとめ

本報告書では、緊急事態救助・復旧活動のための Oppsys を用いたレジリエントシステムを提案した。複雑システム内に Oppsys を含めることで、システムに復旧能力（self-healing）を与え、小さな故障と壊滅的な故障を分ける復帰不能点を過ぎ去る前に、Oppsys は復旧オペレーションを始める。また、Oppsys のためのシステム復旧手法を提案した。確率的なランダムグラフ理論を用いたシステム復旧問題を定式化した。また、Oppsys によるリソースの割当・再配置・解放アルゴリズムを開発した。また、Oppsys 仮想マシンを開発するために、緊急時アプリケーションシナリオ例、通信シーケンス、プリミティブと擬似コードの設計を行った。

Oppsys の可能性は、Helper ノードがそれらと共にもたらす豊富なリソースを利用することで実現される。コンピュータ装置やシステムのパーベイシブがインフラ、建物、乗物、家電などにおいて増加するのに伴い、

Helper 候補の存在は劇的に増加し続けると考える。また、より人口密度の高いエリアでは、より多くの潜在的な Helper ノードが期待できる。

今後の課題として、提案のリソース制御アルゴリズムを Java ベースのフリーの JGraphT library [14]に実装し、詳細なシミュレーション分析を行う。また、本設計に基づいて Oppsys 仮想マシンを実装し、アプリケーションシナリオ例を用いた評価を行う。

【参考文献】

[1] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller and P. Smith "Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines," *Elsevier Computer Networks*, Special Issue on Resilient and Survivable Networks, vol.54(8), Jun. 2010, pp. 1245-1265.

[2] T. Davis, "A Failure of Initiative: The Final Report of the Select Bipartisan Committee to Investigate the Preparation for and Response to Hurricane Katrina," *U.S. House of Representatives*, Feb. 2006, pp.109–364.

[3] K. Kobayashi, "Restoration Status for Damage Caused by the Great East Japan Earthquake and Future Responses," *IEICE Communications Society GLOBAL NEWSLETTER*, Special issue: "Recovery and Reconstruction from Great East Japan Earthquake," Vol. 35(4), Dec. 2011, pp.4-6.

[4] J.-C. Laprie, "From dependability to resilience," *38th IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN 2008)*, (Fast Abstracts), Jun. 2008, pp.G8–G9.

[5] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Commun. ACM*, vol.53(10), Oct. 2010, pp.72-82.

[6] K.M. Sim, "Grid Resource Negotiation: Survey and New Directions," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, vol.40(3), May 2010, pp.245-257.

[7] B.P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach," *Journal of Grid Computing*, vol.9(1), Mar. 2011, pp.3-26.

[8] A. Boukerche, B. Turgut, N. Aydin, M.Z. Ahmad, L. Bölöni, and D. Turgut, "Routing protocols in ad hoc networks: A survey," *Computer Networks*, vol.55(13), Sep. 2011, pp.3032-3080.

[9] T. DeMarco, "Software Engineering: An Idea Whose Time Has Come and Gone?," *IEEE Software*, vol. 26(4), Jul./Aug. 2009, pp. 96, 95.

[10] L. Lilien, A. Gupta, Z. Kamal, and Z. Yang, "Opportunistic Resource Utilization Networks—A New Paradigm for Specialized Ad Hoc Networks," *Computers and Electrical Engineering*, Special Issue: "Wireless Ad Hoc, Sensor and Mesh Networks," Vol.36(2), Mar. 2010, pp.328-340.

[11] B. Bollobas, "Random Graphs," 2nd Edition, *Cambridge Studies in Advanced Mathematics*, vol.73, Cambridge University Press, New York, 2001.

[12] C. Patvardhan, V.C. Prasad and V.P. Pyara, "Generation of K-Trees of Undirected Graphs," *IEEE Trans. on Reliability*, vol.46(2), Jun. 1997, pp.208-211.

[13] Y.R. Sun and W.Y. Zhou, "An Inclusion-Exclusion Algorithm for Network Reliability with Minimal Cutsets," *American Journal of Computational Mathematics (AJCM)*, vol.2(4), Dec. 2012, pp.316-320.

[14] JGraphT, available at: http://jgrapht.org/.

[15] M. Al-Kuwaiti, N. Kyriakopoulos and S. Hussein, "A Comparative Analysis of Network Dependability, Fault-tolerance, Reliability, Security, and Survivability," *IEEE Communications and Surveys & Tutorials*, Vol.11(2), 2009, pp.106-124.

[16] M. Tortorella, "Path Sets and Cut Sets in System Reliability Modeling," Encyclopedia of Statistics in Quality and Reliability, Mar. 2008.

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
Implementation of a Peer-to-Peer-type SIP Client Application on a MANET Emulator	IEEE Region 10 Conference (TENCON), pp.1-6	November, 2012
A System Recoverability Scheme for Resiliency Based on Opportunistic Resource Utilization Systems for Emergency Rescue and Recovery Operations	IEEE International Conference on Technologies for Homeland Security (HST), (submitted)	November, 2013