計算機アーキテクチャ　93－3
（1992. 3. 23）

リスポンシブシステムと通信プロトコル

角田　良明　　菊野　亨

大阪大学　基礎工学部　情報工学科

〒560 大阪府豊中市待兼山町１－１

あらまし：従来、並列･分散システム技術、フォールトトレランス技術、リアルタイムシステム技術が独立して発展してきた。ところが、情報通信システムが大規模、複雑化するにつれて、並列･分散システム環境下でフォールトトレランス機能とリアルタイムシステム機能を統合したリスポンシブシステムの要求が高まっている。本稿では，リスポンシブシステムの基本的な考え方，定義について紹介する．また，フォールトトレラントシステムとリアルタイムシステムの設計法と比較しながらリスポンシブシステムの設計法とその設計例を紹介する．リスポンシブシステムの通信プロトコルへの応用として，異常状態から正常状態へのリアルタイム回復機能をもつリスポンシブプロトコルの形式的な定義を与え，その設計法と検証法を提案する．

Responsive Systems and Communication Protocols

Yoshiaki Kakuda and Tohru Kikuno

Department of Computer and Information Sciences
Faculty of Engineering Science, Osaka University

1-1, Machikaneyama-cho, Toyonaka-shi, Osaka 560, Japan
{kakuda, kikuno}@ics.osaka-u.ac.jp

Abstract: Previously, techniques for parallel and disatributed systems, fault-tolerant and real-time systems have been independently developed. As information and communication systems become large and complicated, responsive systems which integrates fault-tolerant and real-time systems in parallel and distributed environments are required. This paper introduces concepts of responsive systems and its design method in comparison with fault-tolerant and real-time systems. As an application of the responsive systems to communication protocols, this paper gives a formal definition of responsive protocols, in which real-time recovery from abnormal states to normal states is performed, and proposes a design method for responsive protocols and its related verification method.

1. まえがき

近年，情報通システムの発展に伴い，システムの規模は増大し，システムの構成要素も密接に結合し，それらの間でリアルタイム処理が行われている．もし，情報通信システムの一部で障害が起こり，それが原因でシステム全体が停止すれば，その社会的影響は計り知れない．こうした状況を回避し，正常な情報通信サービスの提供を維持するためには，フォールトトレラント技術が必須である．特に，障害回復に時間がかかっては，その間正常な情報通信サービスの提供が滞ってしまい，結局多大な損害を与えてしまう．従って，障害からのリアルタイム回復処理技術の開発が重要な研究課題となっている．

1990年，テキサス大学オースチン校の Malek教授は，上述したようなフォールトトレランス機能とリアルタイム機能の両者を統合した機能を持つシステムをリスポンシブシステム(Responsive System)[8,9]と定義している．また，フォールトトレラントシステムの設計法とリアルタイムシステムの設計法を比較して，リスポンシブシステムの設計法のあるべき姿を示唆している．

本稿では，Malek 教授の示唆したリスポンシブシステムの設計法と通信プロトコル[7]への応用について述べる．

2. リスポンシブシステム

2.1 リスポンシブシステムの定義

リスポンシブシステムは，フォールトトレランス機能とリアルタイム機能の両者を統合した機能を持つシステムと定義される．つまり，フォールトが存在しても期待された情報通信サービスをタイムリーに実行するシステムといえる．リスポンシブシステムの本質は，フォールトが存在した場合における情報通信サービスの実行の予測可能性にあるといえる．

2.2 フォールトトレラントシステムの設計法

要素にフォールトが存在してもシステムに障害が起きないという特性を備えたシステムをフォールトトレラントシステム[10]という．文献［8,9］で示された設計手順は，次の通りである．

（1）フォールトトレランスのためのシステム仕様を定める．この仕様において，フォールトクラス(fault class)，フォールトレイテンシ(fault latency)，フォールトインパクト(fault impact)を明確にする．

（2）システムの弱点を明確にする．それによる潜在的ダメージを評価する．

（3）フォールト／誤り検出(fault/error detection)の技術を開発する．これらの技術の時間(time)，空間(space)，フォールトカバレッジ(fault coverage)を評価する．

（4）フォールト診断(fault diagnosis)，フォールト隔離(fault isolation)の技術を開発する．これらの技術の時間(time)，空間(space)，フォールトカバレッジ(fault coverage)を評価する．

（5）システム回復(recovery)／再構成(reintegration)／再起動(restart)の技術を開発する．

（6）フォールトトレランスの度合を評価する．

（7）仕様の要求を満たすか否かを判定して，満たすまで（2）～（6）を繰り返す．

2.2 リアルタイムシステムの設計法

リアルタイムシステムは，決められた期間あるいは時間内に情報通信サービスの実行を完了することが要求されているシステムをいう．文献［8,9］で示された設計手順は，次の通りである．

（1）タイミングが重大なタスクを明確にし，それに係わる仕様(deadlines, durations, frequency, periodicity)を定める．システムの負荷と環境を特徴付ける．

（2）システム（ハードウェアとソフトウェア）のタイミングを特徴付ける．

（3）（1）で定めた仕様を（2）のシステムのタイミングに写像する．（そのために，最適なリソース割り当て法，スケジューリング法を見つける．）

（4）仕様が満たされているか否かを検証する．

（5）仕様が満たされていなければ，（1）～（4）を繰り返す．

2.3 リスポンシブシステムの設計法

リスポンシブシステムは，デッドラインを越えたものをフォールトと考えるフォールトレラントシステムとして設計するのではなく，フォールトへの対処も仕様の一部と考えるリアルタイムシステムとして設計すべきである．そのため，特定のフォールトのクラスに対しては，フォールト診断，フォールト回復の時間を考慮する必要がある．そうすることによって，正確なデッドラインの検証を行うことができる．更に，パフォーマンスの低下を最小限に停めて冗長性を加える方法の開発も必要となる．

2.4 リスポンシブシステムの設計例

文献［8,9］では，リスポンシブシステムの設計例をいくつか与えている．ここでは，ハイブリッドアルゴリズムアプローチについて示す．文献［10］では，最適化問題の代表例である巡回セールスマン問題を２種類のアルゴリズム，シミュレーテッドアニーリングとタブーサーチを組み合わせて解こうとする設計例を示している．この設計例は，これら２種類のアルゴリズムならびに各種パラメータの異なるバージョンを複数のプロセッサに実装し，並列に実行させ，途中の幾つかのチェックポイントで停止し，それまでの解のなかで最適解に近いものを採用し，その解を入力として全てのプロセッサでの実行を再開するものである（図１参照）．このように設計することにより，最適解に速くたどりつくことができることを実験的に実証している．更に，このアプローチを拡張して，各部分ネットワーク毎に複数のプロセッサで解を求め，それらを組み合わせて全体のネットワークの解を求める分割統治法も導入している．このような拡張を加えることにより，あるプロセッサが故障してもそのプロセッサの仕事を他のプロセッサで肩代りすることにより，リアルタイムな故障回復を実現している．

3. リスポンシブプロトコル

3.1 リスポンシブプロトコルの定義

異常状態から正常状態への回復をリアルタイムで実行する機能を持つ通信プロトコルをリスポンシブプロトコル[4]という．

リスポンシブプロトコルの形式的定義は次の通りである．

［定義１］エラーイベントを原因としない状態遷移の系列を通して初期グローバル状態から到達するグローバル状態を正常状態と定義する．初期グローバル状態から正常

状態への状態遷移系列を正常系列という．一方，正常状態ではないグローバル状態を異常状態と定義する．最初の状態遷移はエラーイベントを原因とし，最終状態を除く全ての中間状態が異常状態であるような状態遷移系列を異常系列という．各異常系列に対して，次の２つの条件を満たしたときプロトコルはリスポンシブであると定義する．条件（１）その系列の最終状態は正常状態である．条件（２）その系列の全ての状態遷移を実行するために必要な時間はあらかじめ与えられたものより小さいか，あるいは等しい．

定義１の条件（１）と条件（２）は，それぞれ通信プロトコルのフォールトトレランスの性質とリアルタイムの性質を表している．

３.２ リスポンシブプロトコルの設計法

異常状態から正常状態へのリアルタイム回復機能を例外処理のルーチンを用いずに実行する通信プロトコルは，自己安定プロトコル(Self-Stabilizing Protocol)[1, 2]と呼ばれ，既にいくつかのプロトコルが提案されている．自己安定プロトコルの利点としては，通信路の遅延，メッセージの転送エラー，プロセスの一時的な故障などによってプロセス間の整合が失われた場合に例外処理ルーチンを用いずに解決できる点や，プロセスの初期化をネットワークのグローバルな状態を把握せずに行なえる点がある．しかし，自己安定プロトコルでは，一般に，信頼性の向上のみが考慮されており，リアルタイム性はあまり考慮されていない．そこで，自己安定性とリアルタイム性の両方の性質を合わせ持つリスポンシブプロトコルが必要になっている．

本稿で提案するリスポンシブプロトコルの設計法は次の通りである．

（１）プロセス構成とプロセス間のメッセージを定める．

（２）正常時の送受信遷移系列を定める．

（３）異常時の送受信遷移系列を定める．

（４）（２）の系列の実行がリアルタイム性を満たしているか否かだけでなく，（３）の系列の実行が，異常状態から正常状態へのリアルタイム回復性を満たしているか否かも検証する．

（５）リアルタイム性が満たされていなければ，（１）～（４）を繰り返す．

３.３ リスポンシブプロトコルの検証

前節で述べたリスピンシブプロトコルの設計法のステップ（４）では，リスピンシブプロトコルの検証が必要となる．その検証では，定義１より，フォールトトレランスの性質とリアルタイムの性質が満たされていることを証明しなければならない．

文献［3］では，プロトコル検証のためのアサイクリック展開法[5,12]とマルチプロセッサシステムのためのタスクスケジューリング法を利用したリスポンシブプロトコルの検証法を提案している．また，ブロードキャスティングプロトコルへの適用例を与えている．

フォールトトレランスの性質については，任意の異常状態から正常状態への遷移系列が存在することを表す述語を時相論理で記述し，それが満たされているか否かをアサイクリック展開法で得られた全ての実行可能遷移系列にそのような遷移系列が含まれているか否かを調べることにより判定する．

リアルタイムの性質については，アサイクリック展開法で得られた全ての実行可能遷移系列が，あらかじめ与えられた時間内にその実行を終了するか否かを，通信プロトコルのプロセスと遷移をそれぞれマルチプロセッサシステムのプロセッサとタスクに対応させたタスクスケジューリング問題に帰着させて解く．文献[6]では，この方法を一般化した検証法を提案している．

図２のネットワークに対しノードv0から全てのノードへメッセージMを伝播するブロードキャスティングプロトコルに適用した場合の各ノード（プロセスに対応）の実行可能な遷移系列の集合を図３に示す．同図において，メッセージAは，メッセージMの確認メッセージである．状態S3を正常状態，それ以外の状態を異常状態とすると，図３は，全ての異常状態から正常状態への遷移系列が存在することを示している．図３の実行可能な遷移系列のうちの１つの遷移系列を表すシーケンスチャートを図４に示す．この遷移系列に対し，タスクスケジューリングアルゴリズムを適用すると，図５のようなタスクグラフが得られる．このグラフは，１遷移に１単位時間かかるとすると，図４で表された遷移系列を実行するのに要する時間は１１単位時間であることを表している．デッドラインを１２単位時間とすると，デッドラインを越えていないことになる．全ての遷移系列に対してデッドラインを越えているかどうかを調べることにより，リスポンシブプロトコルの検証を行うことができる．

４．International Workshop on Responsive Computer Systems

1991年10月3～4日フランスのGolfe-Juanで第１回International Workshop on Responsive Computer Systemsが開催された．イギリス，フランス，アメリカ，日本，オーストリア，ドイツ，イタリア，オランダ，ポルトガルから31名が招待されて参加した．４つのセッションテーマ（１）計算モデル，設計アプローチ，（２）並列処理，フォールトトレランス，リアルタイム処理，（３）スケジューリング，（４）ケーススタディ，に沿って21件の発表があった．この会議では，リスポンシブシステムの概念が新しいため，その形式的な定義の必要性が指摘されていた．まず，各アプリケーション毎の定義を行った後，リスポンシブシステムの一般的な定義がなされるであろう．

第２回は，1992年10月1～2日KDD研究所で開催される予定である．広く参加者を集めるため，今回は論文を公募して行う．リスポンシブシステムの研究は，第一段階にあり，今後の研究の活発化が予想される．

５．あとがき

本稿では，フォールトトレランスとリアルタイムシステムを統合したリスポンシブシステムとその設計法について紹介した．また，通信プロトコルへの応用として，リスポンシブプロトコルを定義し，その設計法を提案した．リスポンシブシステムには，仕様記述，言語，アルゴリズムなどの基本的考察ならびに様々な分野への適用など解決すべき課題は多い．

謝辞

第２回International Workshop on Responsive Computer Systemsの開催に対し，御支援を頂いている（財）国際コミュニケーション基金，（財）情報科学国際交流財団，国際電信電話株式会社（KDD）研究所に感謝致します．

リスポンシブプロトコルの研究に対して有益なコメン

トを頂いた テキサス大学オースチン校Malek教授に感謝致します．また，本発表の機会を与えて頂いた実時間ワークショップの関係者，特に，電子技術総合研究所，戸田賢二氏に感謝致します．

文献

[1] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," Communications of the ACM, 17, 11, pp.643-644, Nov. 1974.

[2] M. G. Gouda and Nicholas J. Multari: "Stabilizing communication protocols," IEEE Trans. Computers, 40, 4, pp.448-458 (April 1991).

[3] Y. Kakuda and T. Kikuno, "Verification of responsiveness for communication protocols," IEICE Japan, Tech. Group Paper FTS91-57, CPSY91-58, Dec. 1991.

[4] Y. Kakuda, T. Kikuno and H. Saito, "A new design method of responsive protocols for communication systems," presented at Int'l Workshop on Responsive Computer Systems, Golfe-Juan, France, Oct. 1991 (to appear in the ONR-Europe report).

[5] Y. Kakuda, Y. Wakahara and M. Norigoe: "An acyclic expansion algorithm for fast protocol validation," IEEE Trans. Soft. Eng., SE-14, 8, pp.1059-1070 (Aug. 1988).

[6] 川島, 角田, 菊野, "述語列のリアルタイム収束性に基づくリスポンシブプロトコルの検証法," 実時間ワークショップ(March 1992).

[7] M. T. Liu : "Protocol engineering," Advances in Computers, 29, pp.79-195 (1989).

[8] M. Malek, "Responsive systems (A challenge for the nineties)," Proc. EUROMICRO'90, 16th Symp. on Microprocessing and Microprogramming, Keynote Address, Amsterdam, The Netherlands, North-Holland, Microprocessing and Microprogramming 30, pp.9-16, August 1990.

[9] M. Malek, "Responsive systems: A marriage between real time and fault tolerance," Fault-Tolerant Computing Systems, Proc. of Tests, Diagnosis, Fault Treatment 5th International GI/ITG/GMA Conference (Nurnberg, Sept. 1991), Springer-Verlag.

[10] M. Malek, M. Guruswamy, H. Owens and M. Pandra:"A hybrid algorithm technique," The University of Texas at Austin, Dept. of Computer Sciences Technical Report TR-89-06, March 1989.

[11] 南谷 崇, "フォールトトレラントコンピュータ," オーム社（平成３年）.

[12] H. Saito, T. Hasegawa and Y. Kakuda : "Protocol verification system for SDL specifications based on acyclic expansion algorithm and temporal logic,"Proc. of 4th Int'l Conf. on Formal Description Techniques (FORTE'91), pp.513-528, Nov. 1991.

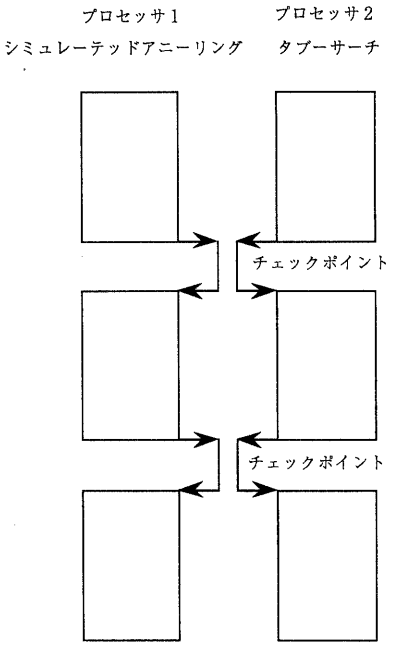


図１　ハイブリッドアルゴリズム

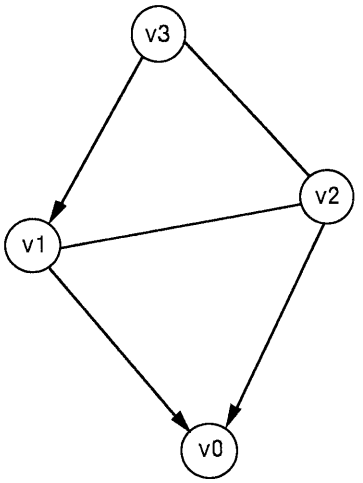
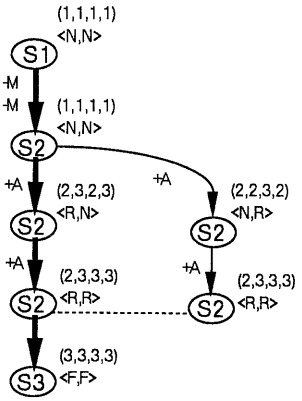
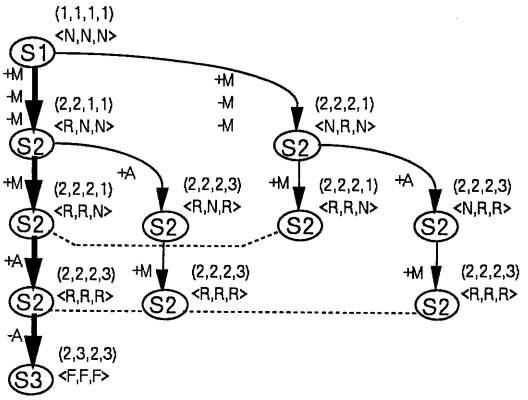


図２　ネットワーク



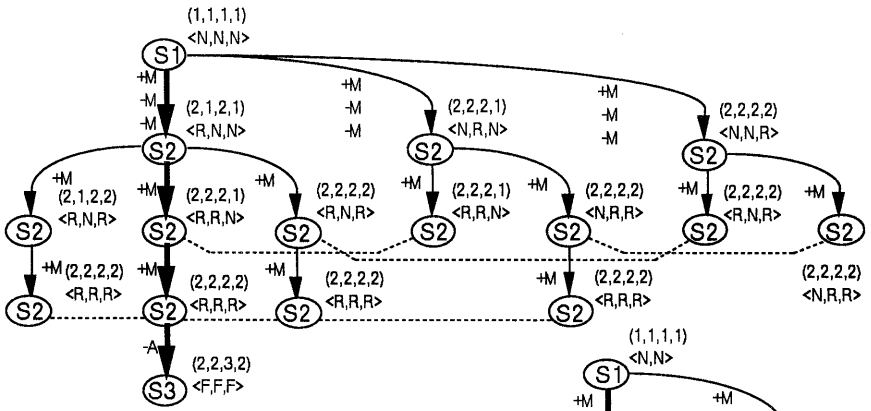
(a,b,c,d) denotes (L0(s,X),L1(s,X),L2(s,X),L3(s,X)) where S1, S2 and S3 are abbreviated as 1, 2 and 3, respectively,
<p,q> denotes <N0(v1),N0(v2)> where NIL, RCVD and FINAL are abbreviated as N,R and F, respectively, and
M and A denote MSG and MSGACK, respectively.

(a) ノードv0



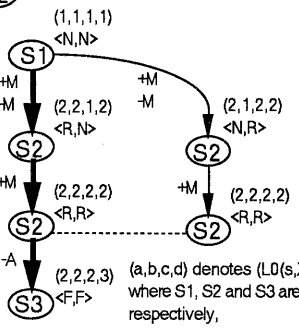
(a,b,c,d) denotes (L0(s,X),L1(s,X),L2(s,X),L3(s,X)) where S1, S2 and S3 are abbreviated as 1, 2 and 3, respectively,
<p,q,r> denotes <N1(v0),N1(v2),N1(v3)> where NIL, RCVD and FINAL are abbreviated as N,R and F, respectively, and
M and A denote MSG and MSGACK, respectively.

(b) ノードv1



(a,b,c,d) denotes (L0(s,X),L1(s,X),L2(s,X),L3(s,X)) where S1, S2 and S3 are abbreviated as 1, 2 and 3, respectively,
<p,q,r> denotes <N2(v0),N2(v1),N2(v3)> where NIL, RCVD and FINAL are abbreviated as N,R and F, respectively, and
M and A denote MSG and MSGACK, respectively.

(c) ノードv2



(a,b,c,d) denotes (L0(s,X),L1(s,X),L2(s,X),L3(s,X)) where S1, S2 and S3 are abbreviated as 1, 2 and 3, respectively,
<p,q> denotes <N3(v1),N3(v2)> where NIL, RCVD and FINAL are abbreviated as N,R and F, respectively, and
M and A denote MSG and MSGACK, respectively.

(d) ノードv3

図3　実行可能な遷移系列の集合

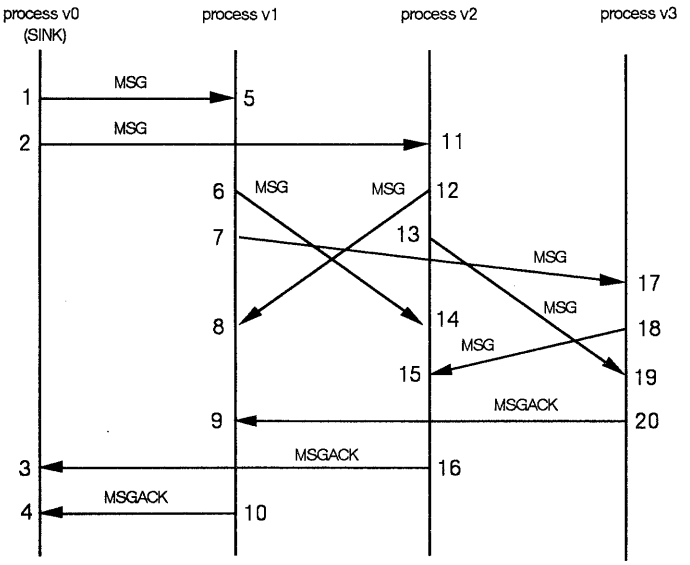


図4　シーケンスチャート

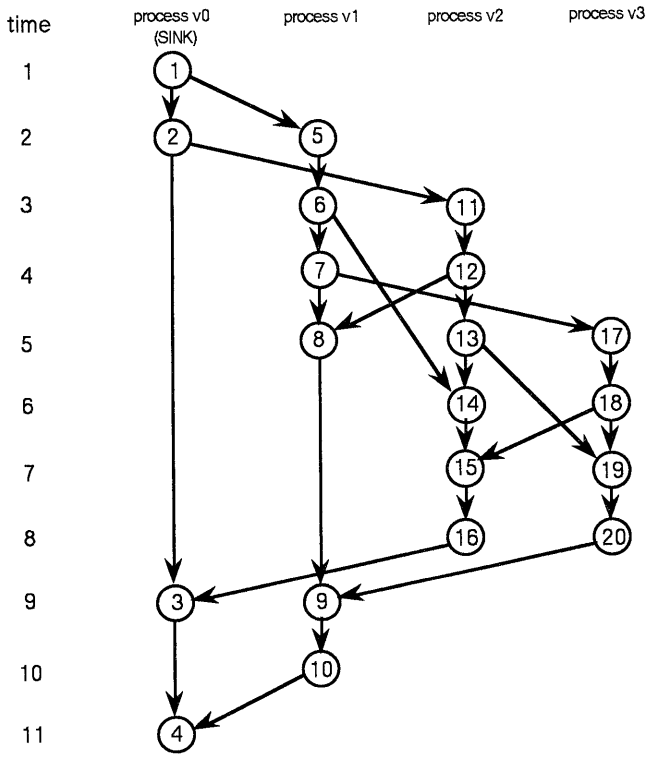


図5　デッドラインの検証