99-04001

大規模コンピュータネットワークの過渡解析に対する効率的算法の開発、並びにネットワーク信頼性を評価するための確率モデルの解析に関する研究

土 肥　　正　　広島大学工学部助教授

１．はじめに

現在、コンピュータネットワークの爆発的な普及により、実時間ネットワークシステムの信頼性を評価するための手法を確立することが急務となっている。特に、ネットワークシステムの障害は、ハードウェア障害よりもむしろソフトウェアによる障害に起因することが多く、ソフトウェアシステムの信頼性を向上させることが最重要課題となっている。ソフトウェアの障害を大別すると、

・ソフトウェアプログラムに含まれる固有フォールトによる障害
・ソフトウェアシステムの経年劣化による障害

に分類される。前者は、ソフトウェアの開発工程中に作り込まれたフォールト（バグ）が運用期間中に表面化することを意味し、後者は、ソフトウェアの運用期間が経過するにつれてソフトウェアシステムの内部構造が変化することにより生じる障害を意味する。このような現象はソフトウェアエージング（software aging ）と呼ばれ、オペレーティングシステムやミッドウェア系システムだけでなく、通信ソフトウェア や Netscape, xrn といった典型的なアプリケーションソフトウェアの運用時においても頻繁に観測されている。そのようなソフトウェアシステムの信頼性・処理性を正確に評価するためには、

(i) システム全体の確率的挙動を描写するためのネットワークモデルの開発
(ii) 構築されたネットワークモデルに基づいたフォールトトレラントシステムに対する設計・管理技術の開発

が必要不可欠となる。ここで、(i) の問題は大規模コンピュータネットワークの過渡解析に対する効率的算法を開発することによって解決され、(ii) の問題は予防保全や冗長構成技術に基づいた信頼性評価法を新たに提案することにより解決されるものと考えられる。

大規模コンピュータネットワークの時間的挙動に関する解析的方法は、ジャクソンネットワークや積解形式をもつ特別な待ち行列ネットワークを除いて、一般には知られていない。また、実際のコンピュータネットワークを待ち行列ネットワークで記述する場合、状態数が極端に増加するため、既存の待ち行列ネットワークの手法を適用することはあまり有効ではない。より広いクラスの確率的離散事象システムを解析するツールとして確率ペトリネットが知られているが、コンピュータシステムの性能評価において局所的に発生する連続状態量を矛盾なく表現するためには、従来の確率ペトリネットが有するモデルの記述能力に限界があることが指摘されていた。そこで、流体ペトリネット（ fluid Petri net ）と呼ばれる新しいクラスのペトリネットを開発し、その効率的算法を提案する。具体的には、系の動的挙動をマルコフ連鎖によって駆動される確率微分方程式によって記述し、離散的逐次近似の手法に基づいてシステムの性能評価指標を算出することを提案する。

次に、通信ソフトウェアに対する信頼性を向上させる目的で、定期的なソフトウェア若化（ rejuvenation ）スケジュールを決定するための確率モデルを提案する。ここでは、通信アプリケーションソフトウェアの時間的挙動を、正常稼働状態、障害が発生可能な状態、障害の発生状態、ソフトウェア若化状態の4状態をもつセミマルコフ過程で記述し、コスト有効性（ cost effectiveness ）と呼ばれるディペンダビリティ評価尺度を新たに導入することにより、ソフトウェア若化スケジュールを解析的に導出する。さらに、一般のソフトウェア運用環境において、統計的に十分な数の障害発生時間データを採集することが事実上困難であることを鑑み、実データからソフトウェア若化スケジュールをノンパラメトリックに推定するためのアルゴリズムを開発する。

2．流体ペトリネットによるシステム解析

2.1 ネットモデルの抽象表現

流体ペトリネットは、時間変数などの連続状態量に依存した離散事象システムを記述・解析するためのペトリネットモデルであり、物理学における流体近似手法の考え方を応用したシステム解析手法として捉えることが出来る。いま、流体ペトリネットを（P, T, A, m_0, F, W, R, G）によって抽象的に記述する。ここで、P は離散プレースの集合P_dと連続プレース（流体プレース）の集合P_c から構成されるプレースの集合、F は流体プレースの数、T はトランジションの集合であり、指数分布に従って発火する時間トランジションの集合T_E と瞬間トランジションの集合T_I から構成される。アークの集合A は 2 つの補集合 A_c とA_dから構成され、A_c は （$P_c \times T_E$）∪（$T_E \times P_c$）として、A_d は離散アークの集合として定義される。m_d =（ # $p_i, i \in P_d$ ）は離散プレース上にあるトークンの数を表すベクトル変数であり、x =（ $x_k, k \in P_c$） を連続プレース上の流体レベルを表すベクトル変数とする。特に、m_d はネット上の離散マーキングと呼ばれ、m を離散マーキングの数を表す変数とする。これより、流体ペトリネットのマーキング集合は（x, m_d）によって表現される。さらに、M は全てのマーキング（ x, m_d ）の集合であり、特に M_d を全ての離散マーキングの集合として定義する。初期マーキングを m_0 =（ x_0, m_{d0}）、離散アークの集合 A_d を {（$P_d \times T$）∪（$T \times P_d$）}×M_d から自然数への写像として表す。

通常の確率ペトリネットと同様に、$G: T \times M \to \{0, 1\}$ を定義する。時間トランジションτ∈ T_E に対して、関数Gはブール関数となる。また、発火率関数 F は時間トランジッション T_E に対して定義される。すなわち、$F: T_E \times M \to R^+$ である。もし時間トランジションτがマーキング m で発火可能ならば、それは発火率 F（ τ, m ）で発火する。いま、重み関数 W が瞬間トランジション T_I に対して定義されるものとする。つまり、$W: T_I \times M_d \to R^+$ とする。もし瞬間トランジション i がマーキング m_d で発火可能ならば、それは確率

$$\frac{W(i, m_d)}{\sum_{\theta \in T_I} W(\theta, m_d)} \tag{1}$$

によって発火する。次に、連続マーキングについて考察すると、流入率（flow rate）R は連続プレースに結合されたアークと時間トランジションに対して定義されるものとしよう。すなわち、$A_c \times M \to R^+ \cup \{0\}$. いま、流体マーキングが時刻 t において $m_t \in M$ であったとすれば、流量（fluid）は率 R（(k, τ), m_t ）でアーク（k, τ）∈ A_c の先のプレース $k \in P_c$ から流出し、同様に、m_tで発火可能な各τ∈T_E に対して、R（(τ, k) ,m_t ）でアーク（ τ, k）∈ A_c の先の連続プレース k に流入する。以上の性質から、マーキング m_t で時刻 t において流体がプレース $k \in P_c$ に流入する瞬間流入率は、

$$r_k(m_t) = \sum_{\tau \in T_E} R((\tau, k), m_t) - \sum_{\tau \in T_E} R((k, \tau), m_t) \tag{2}$$

となる。あらゆる離散マーキング m_d とアーク（τ, k）に対して、上述の瞬間流入率 R（(τ, k), (x, m_t)）は x の区分的連続な関数となる。m_t は連続状態レベルを表す変数 x を含むので、瞬間流入率は例え m_t の離散部分が変化しなかったとしても、時間変数 t の関数として一意に変換する事が出来る。

具体的に、X_k（t ）を連続プレース $k \in P_c$ において時刻 t での流体レベルを表す確率過程として定義する。流体プレースには上限があると仮定出来れば、X_k（t ）≤ B_k なる上限値を設定することが出来る。このとき、確率過程 X_k（t ）のサンプルパスは、よく知られたサンプルパス解析の結果を応用することにより、

$$\frac{dX_k(t)}{dt} = \begin{cases} [r_k(m_t)]^+ & \text{if } X_k(t) = 0 \\ [r_k(m_t)]^- & \text{if } X_k(t) = B_k \\ r_k(m_t) & \text{if } 0 < X_k(t) < B_k \text{ and } r_k(m_{t-}) r_k(m_{t+}) \geq 0 \\ 0 & \text{if } 0 < X_k(t) < B_k \text{ and } r_k(m_{t-}) r_k(m_{t+}) < 0. \end{cases} \tag{3}$$

のような関係を満たすことがわかる。上述の結果は連続プレースの状態変化を表す微分方程式であり、固定されたサンプルパスに対してプレースへの流入と流出関係を記述するための方程式でもある。この関係式をベースに、以下では確率過程 X（t ）=[X_k（t ）, $k \in P_c$] を導入し、ペトリネット上のプレースの状態を表現する2変量確率過程（X（t ）, M_d（t））の確率的性質について考察を行う。

2.2 解析

時刻 $t > 0$ での k 番目のプレース上の流体レベルを確率過程 X_k（t ） によって定義する。前節で定義した流体ペトリネットの離散状態部分に対する可達グラフは状態空間 M の連続状態マルコフ連鎖に帰着出来る。ペトリネット上の

離散状態部分が流体プレースに従属するならば、系を支配するダイナミクスはマルコフ連鎖によって駆動される確率微分方程式によって記述される。ここでは、大規模コンピュータネットワークの信頼性・処理性解析において通常よく仮定されているように、ペトリネット上の離散状態部分が流体プレースのレベル $X_k(t)$ と独立である場合の結果について述べる。S を離散状態空間とし、$Q(x)=[q_{ij}(x)]$ を発火率関数 F から生じる確率推移行列とし、M_d における離散マーキングの集合に対応するものとする。また、プレース $k \in P_c$ は対角行列 $R_k(x)=\mathrm{diag}(r_k(x, m_d)), m_d \in S$ を構成する。いま、確率分布関数 $H(t, x, m_d) = \Pr\{X(t) \le x, M_d(t) = m_d\}$ と確率ベクトル $H(t, x)=[H(t, x, m_d), m_d \in S]$ を定義する。

もし、関数 $R_k(x)$ が x で微分可能、推移率 $Q(x)$ が x の区分的に連続な関数であり、かつ連続状態量を計測する流体プレースの容量に制約がない（無限バッファモデルを仮定する）ならば、2変量確率過程（$X(t)$, $M_d(t)$）は非負の要素をもつ全ての x と $m_d \in S$ に対して確率密度関数 $h(t, x, m_d)$ をもつ。さらに、x が少なくとも一つ 0 の要素をもつならば、以下のような確率関数 $c(t, x, m_d)$ をもつことが示される。

$$c(t,x,m_d) = \frac{\Pr\{X_k(t) = 0 \text{ if } x_k = 0,\ X_k(t) \in (x_k, x_k + dx_k) \text{ if } x_k > 0\}}{\prod_{k;x_k>0} dx_k}. \tag{4}$$

さらに、$h(t, x)$ を $h(t, x, m_d)$ の列ベクトルとすれば、次のような結果を得ることが出来る。

定理 1 : 流体ペトリネットの時間的挙動を記述する状態方程式は以下のようになる。

$$\frac{\partial h}{\partial t} + \sum_k \frac{\partial (hR_k(x))}{\partial x_k} = hQ(x), \tag{5}$$

$$c(t,x,m_d) = 0 \ \text{ if, for any } k,\ x_k = 0 \text{ and } r_k(x,m_d) > 0, \tag{6}$$

$$\begin{aligned} \frac{\partial}{\partial t}c(t,x,m_d) + \sum_{k;x_k=0} h(t,x,m_d) r_k(x,m_d) \quad &+ \quad \sum_{k;x_k>0} \frac{\partial}{\partial x_k} h(t,x) r_k(x,m_d) \\ &= \quad \sum_{i \in S} c(t,x,i) q_{i,m_d}(x), \end{aligned} \tag{7}$$

$$r_k(x,m_d) < 0 \text{ if, for any } k,\ x_k = 0. \tag{8}$$

定理 2: 累積確率分布関数 H が満足するコルモゴロフ方程式は

$$\frac{\partial H}{\partial t} + \sum_k \frac{\partial}{\partial x_k}(HR_k(x)) = HQ(x) - \int_0^{x_F} \cdots \int_0^{x_1} H \cdot \frac{\partial Q(x)}{\partial x_k} dx_1 \cdots dx_F. \tag{9}$$

ここで、上記の偏微分方程式の境界条件は、

$$H(t,x,i) = 0 \text{ at } x_k = 0 \text{ if } r_k(x,i) > 0, \tag{10}$$

および

$$\lim_{x_k \to \infty} \frac{\partial H}{\partial x_k} = 0 \tag{11}$$

となる。

上述の定理より、連続状態量を含む一般化された離散事象システムの確率法則を支配する偏微分方程式が導出されたので、これを離散スキーム上で数値的に解くことによって、一般の大規模ネットワークシステムの過渡解を計算することが可能となる。

3．通信システムの信頼性解析

ソフトウェアエージングによる障害は一過性の障害（ transient failure ）であることが多い。すなわち、障害が発生した後、若干異なる内容（データ、環境）でシステムをリトライすることにより、あたかも障害が発生していなかったかのような操作可能状況に復帰する可能性がある。反面、このような一過性の障害は、ソフトウェアシステムのソースコード上で障害の原因を特定することが極めて困難であることから、その対処法について数多くの研究がなされてきた。特に、ソフトウェア若化（ software rejuvenation ）と呼ばれる方策は、ソフトウェアエージングによる一過性の障害を予防するための有効な方法として認識されており、ソフトウェアシステムの稼働を一時的に停止し、その内部構造を浄化した後にシステムを再稼働する一連の予防保全手続きを意味する。ここで、ソフトウェアシステムの内部構造の浄化とは、ガーベージコレクション（ garbage collection ）やオペレーティングシステムにおけるカーネルテーブル（ operating system kernel tables ）の洗浄（ flushing ）、データ構造の初期化（ reinitializing ）等を示す。アプリケーションシステムの運用上、極端であるが最も頻繁に行われているソフトウェア若化の一例として、ハードウェアリブート（ hardware reboot ）が挙げられる。ここでは、典型的な通信アプリケーションソフトウェアの時間的挙動をセミマルコフ過程によって表現し、各状態推移を表す時間変数が一般分布に従う場合について考える。さらに、アベイラビリティと運用期待費用の2つの評価尺度を統合したコスト有効性と呼ばれる評価尺度について考察を行う。さらに、最適なソフトウェア若化スケジュールを統計的に推定するためのノンパラメトリックアルゴリズムを提案する。提案された方法の利点として、障害発生時間データに理論分布を仮定し、推定・検定に基づいた分布のあてはめを行う必要がなくなることが挙げられる。

3.1 モデルの記述

通信アプリケーションソフトウェアの時間的挙動は、正常稼働状態（状態 0)、障害が発生可能な状態（状態 1)、障害の発生状態（状態 2)、ソフトウェア若化状態（状態 3）の4状態をもつセミマルコフ過程に従って推移するものと仮定する。いま、システムが正常状態から障害発生可能な状態に推移する時間を非負で連続な確率変数 Z によって表現し、その確率分布関数を $\Pr\{Z \le t\} = F_0(t)$、平均を μ_0（>0）とする。システムが障害発生可能な状態に推移すると、ソフトウェア若化を行うか否かの決定をするものとしよう。システムが障害発生可能な状態から具体的に故障に至るまでの時間は、非負で連続な確率変数 X によって記述され、その確率分布関数を $\Pr\{X \le t\} = F_f(t)$、平均を λ_f（>0）とする。一旦障害が発生すると、事後保全が直ちに開始される。ここで事後保全とは、障害が発生した後に事後的にシステムを若化することを意味し、本稿では修理という言葉によって代用する。修理に要する時間 Y もまた非負の連続形確率変数であり、その確率分布関数を $\Pr\{Y \le t\} = F_a(t)$、平均を μ_a（>0）とする。修理が完了すると、システムの障害発生率は正常稼働状態における初期状態まで復旧される。

他方、ソフトウェアの予防的な若化は、システムが障害発生可能な状態に推移した後の T 単位時間経過後になされるものとする。ここで、T は非負で連続な確率変数であり、その確率分布関数を $F_r(t)$、平均を t_0（>0）とする。システム障害が発生する前に時刻 T が経過した場合は、直ちに予防的に若化を開始し、そのシステムオーバーヘッドに対する確率分布関数を $\Pr\{Y \le t\} = F_c(t)$、平均を μ_c（>0） とする。修理の場合と同様に、予防的若化が完了すると、システムの障害発生率は正常稼働状態における初期状態まで復旧される。このモデルは、状態 0 から再度状態 0 に戻るまでの時間間隔を周期にもつセミマルコフ過程であり、すべての状態が再生点（ regeneration points ）となることから、推移確率を求めるために通常のマルコフ解析の手法が適用可能である。特に、障害発生可能な状態から予防的に若化を実施するまでの時間 T が一定である（ periodic rejuvenation ）とすれば、確率分布関数 $F_r(t)$ を次のようなユニット関数

$$F_r(t) = U(t - t_0) = \begin{cases} 1 & \text{if } t \ge t_0 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

に置き換えればよい。

$\overline{F}_f(\cdot) = 1 - F_f(\cdot)$ を障害発生時間分布の余関数、障害発生時の単位時間当たりの修理費用を c_s（>0)、予防的に若化を行う場合の単位時間当たりの費用をc_p（>0）vとすれば、システムにおけるコスト有効性は

$$E_1(t_0) = \lim_{t\to\infty} \frac{\mathrm{E}[\text{operative time on } [0,t)]/t}{\mathrm{E}[\text{cost on } [0,t)]/t} = \frac{\mu_0 + \int_0^{t_0} \overline{F}_f(t)dt}{c_s\mu_a F_f(t_0) + c_p\mu_c \overline{F}_f(t_0)} \tag{13}$$

となり、問題はコスト有効性 $\mathrm{E}_1(t_0)$ を最大にするソフトウェア若化スケジュール t_0^* を決定することになる。コスト有効性は、単位費用当たりの期待稼働可能時間を意味し、システムのアベイラビリティと、予期されないシステムのダ

ウンコスト（重要度に関する重み）と計画的にシステムをダウンさせる際に生じるコストのトレードオフを同時に考慮した信頼性評価尺度となっている。

次に、上述の基本モデルを修正したモデルについて述べる。上述のモデル１では、障害が発生した後に直ちに修理を開始し、修理完了後にシステムは元の状態に復帰するものと考えていた。しかしながら、修理の内容が予防的に実施される若化と異なる場合、すなわち、ファイル系システムにおいて修理が障害により失われたデータを復旧することを含むような状況を考えるのであれば、データを回復した後に再度システム自体の若化を実施する必要がある。よって、モデル 2 では、時間 $\min\{Z+t_0, Z+X+Y\}$ 経過後にソフトウェア若化を実施することになる。このとき、モデル 2 に対するコスト有効性は

$$E_2(t_0)=\frac{\mu_0+\int_0^{t_0}\overline{F}_f(t)dt}{c_s\mu_a F_f(t_0)+c_p\mu_c} \tag{14}$$

となる。

3.2 解析

前述した 2 つのモデルに対して、各々のコスト有効性を最大にする最適ソフトウェア若化スケジュールを導出する。関数 $E_i(t_0)$（i =1, 2）は t_0 に関して連続であるので、t_0 で直接微分することによって、

$$dE_1(t_0)/dt_0 = q_1(t_0)\overline{F}_f(t_0)/\{c_s\mu_a F_f(t_0)+c_p\mu_c\overline{F}_f(t_0)\}^2, \tag{15}$$

$$dE_2(t_0)/dt_0 = q_2(t_0)\overline{F}_f(t_0)/\{c_s\mu_a F_f(t_0)+c_p\mu_c\}^2 \tag{16}$$

を得る。ここで、関数 $q_i(t_0)$（i=1, 2）は、

$$q_1(t_0) = \{c_s\mu_a F_f(t_0)+c_p\mu_c\overline{F}_f(t_0)\}-(c_s\mu_a-c_p\mu_c)\left\{\mu_0+\int_0^{t_0}\overline{F}_f(t)dt\right\}r_f(t_0) \tag{17}$$

$$q_2(t_0) = \{c_s\mu_a F_f(t_0)+c_p\mu_c\}-c_s\left\{\mu_0+\int_0^{t_0}\overline{F}_f(t)dt\right\}r_f(t_0) \tag{18}$$

によって表される関数であり、関数 $r_f(\cdot)$ は確率分布関数 $F_f(\cdot)$ の故障率を表す。

定理 3: モデル１において、$c_s\mu_a > c_p\mu_c$、すなわち、修理に要するコストは予防的にソフトウェアを若化するコストよりも大きいと仮定する。

(1) 障害発生時間分布 $F_r(t)$ が強意 IFR（ Strictly Increasing Failure Rate ）であるとする。
$E_1(0) < \{(c_s\mu_a-c_p\mu_c)r_f(0)\}^{-1}$ かつ $E_1(\infty) > \{(c_s\mu_a-c_p\mu_c)r_f(\infty)\}^{-1}$ ならば、$q_1(t_0^*)=0$ を満たす有限で唯一の最適ソフトウェア若化スケジュール t_0^*（ $0 < t_0^* < \infty$ ）が存在し、そのときの最大コスト有効性は $E_1(t_0^*)=\{(c_s\mu_a-c_p\mu_c)r_f(t_0^*)\}^{-1}$ となる。
一方、$E_1(0) \geq \{(c_s\mu_a-c_p\mu_c)r_f(0)\}^{-1}$ もしくは $E_1(\infty) \leq \{(c_s\mu_a-c_p\mu_c)r_f(\infty)\}^{-1}$ ならば、最適ソフトウェア若化スケジュールはそれぞれ $t_0^*=0$ および $t_0^*\to\infty$ となり、最大コスト有効性は $E_1(0)=\mu_0/c_p\mu_c$ および $E_1(\infty)=(\mu_0+\mu_f)/c_s\mu_a$ となる。

(2) 障害発生時間分布 $F_r(t)$ が DFR（ Decreasing Failure Rate ）であるとする。
このとき、$c_s\mu_a\mu_0 > c_p\mu_c$（ $\mu_0+\mu_f$） ならば $t_0^*=0$ となり、そうでなければ $t_0^*\to\infty$ となる。

定理 4: モデル2において、

(1) 障害発生時間分布 $F_r(t)$ が強意 IFR（ Strictly Increasing Failure Rate ）であるとする。
$E_2(0) < \{c_s\mu_a r_f(0)\}^{-1}$ かつ $E_2(\infty) > \{c_s\mu_a r_f(\infty)\}^{-1}$ ならば、$q_2(t_0^*)=0$ を満たす有限で唯一の最適ソフトウェア若化スケジュール t_0^*（ $0 < t_0^* < \infty$ ）が存在し、そのときの最大コスト有効性は $E_2(t_0^*)=\{c_s\mu_a r_f(t_0^*)\}^{-1}$ となる。
一方、$E_2(0) \geq \{c_s\mu_a r_f(0)\}^{-1}$ もしくは $E_1(\infty) \leq \{c_s\mu_a r_f(\infty)\}^{-1}$ ならば、最適ソフトウェア若化スケジュールはそれぞれ $t_0^*=0$ および $t_0^*\to\infty$ となり、最大コスト有効性は $E_1(0)=\mu_0/c_p\mu_c$ および $E_2(\infty)=(\mu_0+\mu_f)/(c_s\mu_a+c_p\mu_c)$ となる。

(2) 障害発生時間分布 $F_r(t)$ DFR（ Decreasing Failure Rate ）であるとする。

このとき、（$c_s \mu_a + c_p \mu_c$）$\mu_0 > c_p \mu_c$（$\mu_0 + \mu_f$）ならば $t_0^* = 0$ となり、そうでなければ $t_0^* \to \infty$となる。

上述の結果において、$t_0^* = 0$ はシステムが障害発生可能な状態に推移した直後に予防的に若化することが最適であり、$t_0^* \to \infty$は予防的にソフトウェアを若化しないことが最適であることをそれぞれ示している。また、モデル 1 において仮定された $c_s \mu_a > c_p \mu_c$ は、システムの予防保全を行う動機を与えるために必要かつ妥当な条件である。一方、モデル 2 では事後的に修理を施した後に再度ソフトウェアの若化を行うため、パラメータに関する条件が最適解を特徴づけるために必要とされないことは興味深い。

3.3 統計アルゴリズムの開発

前章で得られた最適ソフトウェア若化スケジュールに関する結果は、パラメータ μ_0, μ_c, μ_a および障害発生時間分布 $F_f(t)$ が既知である状況下では有効である。しかしながら、実際のソフトウェアシステムの運用環境において、上記のパラメータに関する統計情報が正確に把握されていることは希である。特に、障害発生時間分布 $F_f(t)$ を特定するためには、かなり多くの障害発生時間に対するデータが必要とされる。また、仮に過去におけるシステムの運用実績からデータが取得されていたとしても、$F_f(t)$ に適当な理論分布を仮定し、複数のパラメータ推定法から最良のものを選択し、さらに理論分布と実データの適合度を検証するという一連の手続きは非常に煩雑であるばかりか、データ解析に関する経験と要領を必要とすることが知られている。ここでは、実際に障害発生時間データが与えられているという状況において、コスト有効性を最大にする最適ソフトウェア若化スケジュールを（理論分布を仮定することなく）ノンパラメトリックに推定するためのアルゴリズムについて考察する。

いま、障害発生時間分布 $F_f(t)$ に対する標準総試験時間変換（ scaled total time on test transform ）を次のように定義する。

$$\phi(p) = (1/\lambda_f) \int_0^{F_f^{-1}(p)} \overline{F}_f(t)dt. \tag{19}$$

ここで、$F_f(t)$ は絶対連続かつ非減少関数であるので、その逆関数

$$F_f^{-1}(p) = \inf\{t_0; F_f(t_0) \geq p\}, \quad 0 \leq p \leq 1 \tag{20}$$

が必ず存在する。よく知られた結果として、確率分布関数 $F_f(t)$ が IFR（DFR）であるための必要かつ十分条件は、関数 $\phi(p)$ が $p \in [0, 1]$に関して凹（凸）関数であることである。これより、コスト有効性を $F_f(t)$ の標準総試験時間変換を用いて書き換えることにより、以下の結果を得る。

定理 5: モデル i（$i = 1, 2$）において、コスト有効性 $E_i(t_0)$ を最大にする最適ソフトウェア若化スケジュールを求める問題は、以下の問題の解 p^*（$0 \leq p^* \leq 1$）を求めることと等価である。

$$\max_{0 \leq p \leq 1} \frac{\phi(p) + \alpha}{p + \beta_i}. \tag{21}$$

ここで、

$$\alpha = \mu_0/\lambda_f, \tag{22}$$

$$\beta_1 = c_p\mu_c(c_s\mu_a - c_p\mu_c), \tag{23}$$

$$\beta_2 = c_p\mu_c/c_s\mu_a. \tag{24}$$

定理 5 から、障害発生時間分布 $F_f(t)$ が既知であるならば、最適ソフトウェア若化スケジュールは $t_0^* = F_f^{-1}(p^*)$ によって求めることが出来る。ここで、p^*（$0 \leq p^* \leq 1$）は 2 次元平面上で点 $(-\beta_i, -\alpha) \in (-\infty, 0) \times (-\infty, 0)$ から曲線 $(p, \phi(p)) \in [0, 1] \times [0, 1]$ に引いた線分のうち、最も大きい傾斜を示す曲線上の点に対する x 座標値 p^*によって与えられる。この結果は、3. で導出した代数的方法を幾何学的に解釈したことに他ならず、定理 5 は定理 3 と定理 4 の双対定理となっている。

次に、障害発生時間分布 $F_f(t)$ は未知であるが、対応する障害発生時間データの順序統計量 $x_1, x_2, \ldots, x_n$ が観測されており、これらは打ち切りのない完全データであると仮定する。障害発生時間分布 $F_f(t)$ の推定量を x_j（$j = 0, 1,$

2, **...**, n ） に基づいた経験分布関数

$$F_n(x) = \begin{cases} j/n & \text{for} \quad x_j \le x < x_{j+1}, \\ 1 & \text{for} \quad x_n \le x \end{cases} \tag{25}$$

によって定義する。さらに、経験分布関数に基づいた標準総試験時間変換の推定量として、次にような標準総試験時間統計量（ scaled total time on test statistics ）を定義する。

$$\phi_{nj} = \psi_j / \psi_n. \tag{26}$$

ここで、関数

$$\psi_j \quad = \quad \sum_{k=1}^{j} (n-k+1)(x_k - x_{k-1}), \quad j = 1, 2, \cdots, n; \ \psi_0 = 0 \tag{27}$$

は総試験時間統計量と呼ばれる。最終的に、点列 （F_n（x）, ϕ_{nj}）（ j =0, 1, 2, **...**, n ）を 2 次元平面上にプロットし、それらを線分で繋ぐことにより、標準 TTT プロット（ scaled total time on test plot ）を得る。
推定量 （F_n（x）, ϕ_{nj}）（ j =0, 1, 2, **...**, n ）は（p, ϕ（ p））($p \in [0, 1]$）のノンパラメトリック推定量であるので、定理 5 の結果を直接適用することにより、最適ソフトウェア若化スケジュールに関する以下の定理を得る。

定理 6: モデル i（ i =1, 2）において、障害発生時間に対するn（>0）個の完全データの順序統計量 $x_1 \le x_2 \le \cdots \le x_n$ が観測されているものとする。

(i) コスト有効性を最大にする最適ソフトウェア若化スケジュールのノンパラメトリック推定量 t_0^* は、次の式を満たす x_{j*} によって与えられる。

$$j^* = \left\{ j \mid \max_{0 \le j \le n} \frac{\phi_{nj} + \alpha}{j/n + \beta_i} \right\}. \tag{28}$$

ここで、式（22）の λ_f はサンプル平均 $\Sigma_{k=1}^{n} x_k/n$ によって置き換えられる.

(ii)（i）で与えられた最適ソフトウェア若化スケジュールのノンパラメトリック推定量は強一致推定量である。すなわち、x_{j*} は $n \to \infty$のとき（未知であるが）真の最適解 t_0^* に確率 1 で漸近収束する。

定理 6 の証明は Glivenko-Cantelli の補題と大数の強法則から証明される。定理 6 より、障害発生時間データが与えられた後に、具体的に確率分布関数を仮定することなく最適ソフトウェア若化スケジュールを推定できることは興味深い。また、提案された推定アルゴリズムはグラフ上で最適解を推定する幾何学的方法であることから、パラメータの感度分析などを視覚的に行うことができる点が特徴となっている。

4. 結論

本報告では、大規模コンピュータネットワークの過渡解析を効率的に行う方法論として、従来の確率ペトリネットに連続状態変数を許容することが可能な流体ペトリネットを提案し、系のダイナミクスを記述する偏微分方程式を導出した。より複雑なシステムに対しては、ここで求められた偏微分方程式が確率偏微分方程式になるため、もはや通常のモンテカルロシミュレーションによる信頼性・処理性評価を行うことが困難であることが理論的に示される。本報告では最も単純な構造をもつ確率的離散事象システムに対する一般的な解法を提案したが、系を支配する確率法則に無記憶性が仮定出来ない場合、マルコフ再生理論（ Markov regenerative theory ）を適用して、導出された偏微分方程式を修正する必要が生じる。しかしながら、そのような場合においても、本報告で述べられた方法は適用可能であるので、確率的離散事象システムの解析ツールとしてはかなり汎用性の高いものになっていることに注目すべきである。上述の方法論に基づいて実用規模のクライアントサーバシステムやクラスターシステムの信頼性・処理性評価を実行した結果、実際のシステム挙動を忠実にモデル化出来ることがわかった。さらに、モンテカルロシミュレーションとは異なり、乱数の精度に依存しない方法であるため、高い次数で信頼性を評価する場合においても誤差の影響が極めて少ないことが、

提案手法の利点として挙げられる。

また、本研究のもう一本の柱であるネットワーク信頼性解析では、ある通信ソフトウェアの運用形態に着目し、コスト有効性を最大にするソフトウェア若化スケジュールを求めるためのセミマルコフモデルについて考察を行った。導出された結果は従来研究の結果を一般化したものであり、ソフトウェアエージングの現象をより詳細に記述することが可能である。さらに、実際の実時間ソフトウェアシステムにおいて、障害発生時間の統計的性質を完全に把握することは容易ではないという点を鑑み、障害発生時間データから最適なソフトウェア若化スケジュールをノンパラメトリックに推定するためのアルゴリズムを開発した。提案された統計アルゴリズムは、障害発生時間データの取得に関してかなり早い段階で漸近的に真の最適解に収束することがわかった。本報告で取り扱ったコスト有効性は、アベイラビリティなどの信頼性評価尺度と異なり、システムを計画的（予防的）にダウンさせる際に生じるコストと障害の発生によって事後的に生じるコストの違いを明確に区別することが可能となる。実際のソフトウェアシステムにおいて障害が発生した場合、その被害の大きさは単にダウン期間の相対的な大きさだけで測ることが困難であることは周知の通りである。本報告では、2種類のシステムダウンに関する重要度の違いをコストパラメータの比 c_s/c_p によって表現したが、社会的に影響力の大きいソフトウェアシステムになればなるほど、そのコストパラメータの比は極端に大きくなるものと予想される。各々のコストパラメータの値はシステムの規模と社会的価値に大きく依存するため、ある程度主観的に設定せざるを得ない反面、コストを定量的に見積もるための方法論を確立することは重要な課題である。

＜ 発　表　資　料 ＞

題　　名	掲載誌・学会名等	発表年月
A simulation study to analyze unreliable file systems with checkpointing and rollback recovery	IEICE Transactions on Fundamentals Electronics, Communication and Computer Sciences	2000年５月
The optimal age-dependent checkpoint strategy for a stochastic system subject to general failure mode	Journal of Mathematical Analysis and Applications	2000年５月
The optimal preventive maintenance policy for a software system with multi server station	6th ISSAT International Conference on Reliability and Quality in Design	2000年８月
Bivariate cumulative Bernoulli trial process and its application to reliability assessment	6th ISSAT International Conference on Reliability and Quality in Design	2000年８月
Bayesian statistical estimation of the optimal shutdown schedule in a computer system	The 2000 Joint Statistical Meetings	2000年８月
Bayesian estimation for a unified software reliability growth model and its comparison	The 2000 Joint Statistical Meetings	2000年８月
Performance analysis of a transaction based software system with shutdown	2nd ACM International Workshop on Software and Performance	2000年９月
Heuristic self-organization algorithms for software reliability assessment and their applications	11th IEEE International Symposium on Software Reliability Engineerring	2000年10月
Analysis of software cost models with rejuvenation	5th IEEE International Symposium on High Assurance Systems Engineerring	2000年11月
The age-dependent optimal warranty policy and its application to software maintenance contract	5th International Conference on Probabilistic Safety Assessment and Management	2000年12月
Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule	IEEE 2000 Pacific Rim International Symposium on Dependable Computing	2000年12月