情報処理学会論文誌 Vol. 51 No. 12 2310–2318 (Dec. 2010)

推薦論文

# アクルーアル故障検出器 ACCMOS の実装と評価

林 原 尚 浩[†1]

広帯域ネットワークの普及や情報家電の登場により各家庭でもホームネットワークを構築し，様々なサービスに依存しつつある．しかし，ホームネットワークでは企業で導入しているようなモニタリングシステムは，設置や監視設定などが複雑なため，一般家庭で導入することは困難である．本研究では，ホームネットワークなどの比較的小規模なシステム管理のためのモニタリングシステムの開発を目的として，その中核となる故障検出器 ACCMOS の実装とその性能評価を行った．ACCMOS は，アクルーアル故障検出方式に基づいて実装され，ネットワークの状況に適応した適切なシステムの監視や監視対象の状態に関する細粒度の情報を提供することができる．性能評価のために 3 つの異なるネットワーク環境で監視対象を ACCMOS が監視し，その際の正検出率，平均誤検出レート，平均故障検出時間を計測した．その結果，いずれの環境でも高い正検出率，低い平均誤検出レートが得られ，高精度の故障検出が実現できることを実証した．

# Implementation and Performance Evaluation of the ACCMOS Failure Detector

Naohiro Hayashibara[†1]

In this paper, I show an implementation of the accrual failure detector model, that I call the ACCMOS failure detector as a core component of a monitoring system for home network. ACCMOS can dynamically adjust to the current network condition, and can adjust to diverse requirements of users simultaneously. It provides information of monitored nodes as the continuous scale, called suspicion level, instead of binary values. I have made several sets of experiments with different environments to evaluate this implementation with a viewpoint of accuracy and detection time.

## 1. はじめに

企業などのサービスを提供しているシステムは，サーバなどの故障によるサービスの停止で多大な損害を被る可能性があるため，管理者を置いて常時，システムの監視を行っており，管理者支援のためのモニタリングシステムも多く開発されている．一方，FTTH などの広帯域ネットワークの普及により，家庭内でも小規模なネットワーク（ホームネットワーク）を構築し，コンピュータなどを接続しているが，ホームネットワークでは企業で導入しているようなモニタリングシステムは，設置や監視のための設定などが複雑なため導入することは困難である．また，このようなモニタリングシステムは機能が豊富である一方で，システム管理の経験がない人にとっては不必要な情報まで提供するため，逆にシステムの状況を理解することが難しくなる場合もある．

ホームネットワークにおいては，将来的に様々なネットワーク家電が導入され，日常生活がそれらのサービスに強く依存することも考えられる．このような場合，家電によるサービスは生活と直結するため，それらの監視，故障の発見は重要性を増すことが考えられる．たとえば，防犯装置がホームネットワークに接続されている場合，防犯装置の故障は，居住者の身の危険を招く可能性もあるため，このようなノードのモニタリングは重要である．

本研究では，ホームネットワーク向けのユーザビリティの高いモニタリングシステムの開発を目的としている（**図 1** 参照）．その第 1 段階として，監視対象ノードをモニタする故障検出器 ACCMOS の実装を行い，性能評価を行った．この故障検出器は，2004 年に林原らによって提案され，近年 Facebook[6] や OurGrid[13]，APPIA[5] などの実用的なサービスやプラットフォームで採用されているアクルーアル故障検出器（Accrual Failure Detectors）[4),8)] に基づいて実装されている．この故障検出器はネットワーク適応性を有することが実証されており[8]，多種多様なホームネットワーク環境での故障検出に向いている．

## 2. 故障検出器

故障検出器は，非同期分散システムにおいて分散合意アルゴリズムや全順序ブロードキャストなどの耐故障分散アルゴリズムを実装するために不可欠なコンポーネントとして用い

†1 京都産業大学コンピュータ理工学部
Faculty of Computer Science and Engineering, Kyoto Sangyo University
本論文の内容は 2009 年 10 月のマルチメディア通信と分散処理ワークショップにて報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．


© 2010 Information Processing Society of Japan

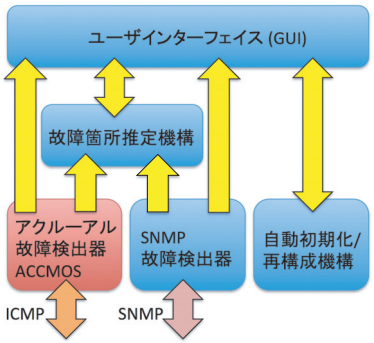



**図 1**　モニタリングシステムにおける ACCMOS の位置づけ
Fig. 1　The architecture of the monitoring system.

られてきた．本章では，既存の故障検出器，アクルーアル故障検出器とそれぞれの実装について述べる．

### 2.1　故障検出器の理論的モデル

#### 2.1.1　非信頼性故障検出器

耐故障分散システムの分野で用いられてきた故障検出器は，監視対象のプロセスの中で，故障である疑いがあるプロセスのリストを出力する一種のオラクルであると定義されている．1996 年に Chandra らによって，非同期分散システムにおける分散合意問題を解決するために必要な故障検出器のクラスとそれらの定義が，非信頼性故障検出器（Unreliable Failure Detectors）として提案された[2]．

#### 2.1.2　アクルーアル故障検出器

アクルーアル故障検出器は，故障検出器の理論的なモデルである[8]．従来の非信頼性故障検出器モデルが非同期システムでは実装不可能であるのに対し，このモデルは実装可能であり，さらに非信頼性故障検出器モデルへの変換アルゴリズムを持つ[4]．これは非信頼性故障検出器を前提としたすべての耐故障分散アルゴリズムに適用可能であることを示し，理論と実装の橋渡しをする役割を担っている．

アクルーアル故障検出器は，監視対象ノード（もしくは，プロセス）の故障している度合いを示す *suspicion level* という値を出力する故障検出器として定義されている．今，監視元ノード $q$ が監視対象ノード $p$ を監視している場合を考える．アクルーアル故障検出器はノード $q$ 上で動作している．$p$ について時刻 $t$ に問合せを行った場合，アクルーアル故障検出器は時刻 $t$ における $p$ の故障している度合いを示す連続的な値を出力する．したがって，ノード $p$ の suspicion level $susp\_level_p(t)$ は以下のように定義できる．ただし，$\mathbb{T}$ は時刻の集合，$\mathbb{R}^+$ は正の実数の集合を示す．

$$susp\_level_p(t) : \mathbb{T} \to \mathbb{R}^+ \tag{1}$$

また，上記で定義した $susp\_level_p(t)$ は以下の性質を満たす．

**Property1（漸近的完全性）**　ノード $p$ が故障していれば，$susp\_level_p(t)$ は時間の増加とともに $\infty$ に近づく．

**Property2（将来的な単調増加性）**　ノード $p$ が故障していれば，ある時刻 $t_m$ が存在し，$t_m$ 以降 $susp\_level_p(t)$ が単調に増加する．

**Property3（上限の存在）**　ノード $p$ が正常であれば，またそのときに限り，$susp\_level_p(t)$ は上限値を持つ．

**Property4（リセット）**　ノード $p$ が正常ならば，$\forall t_0 \in \mathbb{T}$ において，$susp\_level_p(t) = 0$ となる時刻 $t \geq t_0$ が存在する．

アクルーアル故障検出器に基づいた実装を行う場合，出力する suspicion level の値は上記の性質を満たす必要がある．しかし，監視対象ノードの監視方法や suspicion level の算出に関しては自由な実装が可能である．

### 2.2　故障検出器の実装

故障検出器の実装には大きく分けて 2 種類存在する．1 つはタイムアウト型故障検出器と呼ばれるもので，パラメータとしてタイムアウト $\Delta_{to}$ を設定し，$\Delta_{to}$ を用いて監視対象ノードの故障の可否を判定し出力する．これは，故障検出器の出力値だけに注目すると，2.1.1 項で述べた非信頼性故障検出器モデルに近い．もう 1 つは，2.1.2 項で述べたアクルーアル故障検出器モデルに基づいた故障検出器である．この故障検出器は監視対象の故障している度合いを示す suspicion level を出力するが，監視を行っている各ユーザやアプリケーションは，故障判定を行うために suspicion level に対するしきい値 $\Phi$ を設定する．たとえば，あるユーザ $A$ が監視対象ノード $p$ を監視している場合，どれくらいの故障している度合いで $p$ を故障と判定するかを $\Phi_p^A$ として持つ．

#### 2.2.1　タイムアウト型故障検出器

一般的な故障検出器の多くは，このタイムアウト型故障検出器に分類される．このタイプの故障検出器は，まず監視対象ノード $p$ の故障を判定するためのタイムアウト値 $\Delta_{to}$ をパラメータとしてユーザ，もしくはプロセスから与えられる．初期状態において，すべての監視対象ノードは *trust* という状態に設定されている．


© 2010 Information Processing Society of Japan

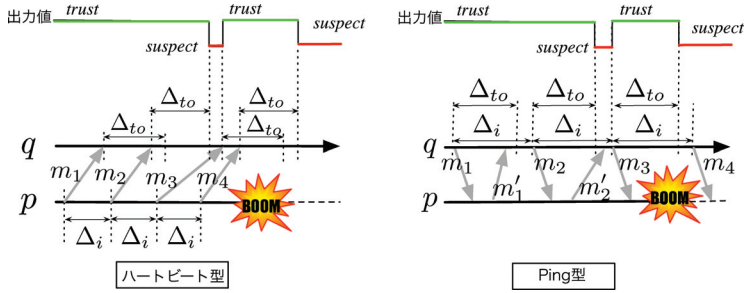



**図 2**　ハートビート型と Ping 型の故障検出器における故障判定とタイムアウト $\Delta_{to}$
Fig. 2　Failure detection mechanism with timeout $\Delta_{to}$ in heartbeat-style and ping-style failure detectors.

その後，一定時間 $\Delta_i$ ごとに $p$ と $q$ の間でメッセージ通信を行い，$p$ からのメッセージが $q$ において $\Delta_{to}$ 以内に受信できない場合，$p$ が故障の疑いがあるため，*suspect* という状態へ移行する．ユーザやプロセスが $p$ の状態を問い合わせた場合，故障検出器はその時点での $p$ の状態 $\{trust, suspect\}$ を返す．**図 2** では，ハートビート型と Ping 型の故障検出器における故障判定を示している．

タイムアウト型故障検出器は，実際に大規模なシステム管理のためのモニタリングシステムに用いられている．それらは，故障検出以外にも，監視対象ノードのサービスの監視や統計情報を可視化するなどの機能を持っているものもある．

以下に，いくつかの代表的なモニタリングシステムをあげる．

#### 2.2.1.1　代表的なモニタリングシステムの実装

モニタリングシステムの実装は，Nagios[11]，Xymon[14]，Hinemos[12] などをはじめとしてとして数多く存在する．特に，オープンソースの Nagios は，多くの機能がプラグインとして実装されているため拡張性が高く．VRML を用いた 3D ネットワーク図の表示などシステムの構成を視覚的に把握しやすい．

#### 2.2.1.2　適応型故障検出器

適応型故障検出器（Adaptive Failure Detectors）は，ネットワークの状態に適応した故障検出を行うため，動的にタイムアウトを調整するアルゴリズムを用いたタイムアウト型故障検出器である．

Chen ら[3] は，独自に定義した QoS パラメータと確率的なネットワークのモデルを入力することで，ユーザの要求とネットワークの状況に適応するアルゴリズムを提案した．Bertier ら[1] は，TCP のフロー制御などに用いられている Jacobson のアルゴリズム[9] を用いてタイムアウトを動的に変化させ，ネットワーク適応性の高い故障検出器を実現した．

これらの適応型故障検出器は，ともに監視対象ノードが送信するハートビートメッセージを用いて監視を行い，タイムアウトによって故障判定をしている．

#### 2.2.1.3　タイムアウト型故障検出器の特徴

タイムアウトを用いたモニタリングシステムや故障検出器における故障検出の精度は，タイムアウトが適切に設定されているかどうかに依存する．しかし，監視対象ノードの所属するネットワークの特性を熟知した管理者の経験や勘がないと，ネットワークの状況に最適なタイムアウトの値を設定することは困難である．また，ネットワークの状況も絶えず変化することが多い．ネットワーク適応型の故障検出器も存在するが，多くのユーザの異なる故障検出に対する要求[⋆1]を同時に実現するためには，それぞれの要求に対応したタイムアウトを管理する必要があり，スケーラビリティに難がある．

### 2.2.2　アクルーアル $\varphi$ 故障検出器

$\varphi$ 故障検出器（$\varphi$ Failure Detector）[8] は，アクルーアル故障検出器モデルに基づいた故障検出器のプロトタイプ実装である．この故障検出器は，故障検出器とユーザやアプリケーションを含めた故障検出に関する一連の処理を以下の 3 つのステップに分けて設計されている．

(1)　**モニタリング**．監視対象ノードと通信を行うことによって，その状態を予測するための情報の収集，解析を行う．
(2)　**インタープリテーション**．モニタリングによって得た情報により，監視対象ノードの故障判定を行う．
(3)　**アクション**．ユーザやアプリケーションが，故障判定を受けて行う動作を指す．

タイムアウト型故障検出器モデルに基づいた多くの実装は，上記のモニタリングとインタープリテーションを故障検出器内部で行っている．特に，インタープリテーションはタイムアウトによって故障判定するため，ボトルネックになり監視対象ノードの情報を要求するユーザやアプリケーションが増えるとパフォーマンスが低下する．

$\varphi$ 故障検出器はこの問題を解決するために，インタープリテーションを故障検出器から切り離し，ユーザやアプリケーションにおいて行うことによってスケーラビリティの向上を

⋆1「故障検出に時間がかかってもよいから誤検出を低減させたい」，「精度よりも故障検出にかかる時間を短くしたい」などの要求が考えられる．


© 2010 Information Processing Society of Japan

図っている．$\varphi$ 故障検出器は，モニタリングにおいて監視対象ノード $p$ から送られてくるハートビートの受信間隔を蓄積する．蓄積したデータが正規分布に従っていると仮定し，正規分布の累積密度関数を用いて時刻 $t_{now}$ における $susp\_level_p(t_{now})$ を表す値 $\varphi_p$ をユーザやアプリケーションに提供する．

ユーザやアプリケーションは個別に $\varphi_p$ に対するしきい値 $\Phi_p$ を設定し，独立して故障判定を行う（たとえば，$\varphi_p > \Phi_p$ となったときに故障と判定）．また，$\varphi_p$ は連続的な値であるため，1 つのアプリケーションに複数のしきい値 $\Phi_p^1, \Phi_p^2, \ldots, \Phi_p^n$ を設定し，$\varphi_p$ の推移に応じて段階的に，監視対象ノード $p$ の故障に備えた処理を行うこともできる．

$\varphi$ 故障検出器は，タイムアウト型故障検出器が時間軸であるタイムアウトを固定して故障判定を行うのに対し，要求する故障検出精度を入力として与え，ネットワークの状況と入力された故障検出精度に合わせた故障検出を行う画期的な故障検出器である．

$\varphi$ 故障検出器と前述の適応型故障検出器との比較実験の結果，ネットワーク適応性に関して同等の性能を持つことが確認された[8]．これによって，$\varphi$ 故障検出器はアクルーアル故障検出器モデルが持つ，ユーザ要求への適応性やスケーラビリティなどの利点に加え，ネットワーク適応性においても高い性能を持つことを示した．

#### 2.2.2.1　アクルーアル故障検出器モデルおよび $\varphi$ 故障検出器の実用例

ソーシャルネットワーキングサービス Facebook[6]，Twitter[18] などで用いられている大規模分散データベース管理システム Cassandra[16] や通信フレームワーク APPIA[5]，グリッドミドルウェアである Our Grid[13] においては透過的なグリッドノードへのアクセスを提供する JIC（Java Internet Communication）[10] などでアクルーアル故障検出器モデルに基づいた故障検出器が実装されている．しかし，これらはそれぞれのプラットフォーム依存の実装であり，一般的なモニタリングシステムとして使用することはできない．

## 3.　ACCMOS の設計と実装

本研究では，ホームネットワークなどの中小規模のシステム向けのモニタリングシステムを開発している．このモニタリングシステムは，初期設定の自動化機構，アクルーアル故障検出器，GUI などのコンポーネントにより構成される．

本論文では，このモニタリングシステムの中核となる故障検出器 ACCMOS の実装と性能評価に焦点を当てる．

アクルーアル故障検出器モデルに基づいて実装された ACCMOS は，従来のタイムアウト型故障検出器が監視対象ノード $p$ の状態を二値（たとえば，$\{trust, suspect\}$）で表現していたのに対し，$\varphi$ 故障検出器同様，suspicion level $\varphi_p$ という連続的な値で表現することによって，より詳細な監視対象ノードの状態を提供することができる．

本章では，一般的なモニタリングシステムの中核となる故障検出器 ACCMOS の実装の詳細について述べる．

### 3.1　監視対象のモニタリング

今，故障検出器 ACCMOS が動作しているノードを $q$ とし，監視対象ノードを $p$ とする．$q$ が $p$ を監視するためには，それらの間で通信を行う必要がある．$\varphi$ 故障検出器ではハートビート型の実装を行っている[8]．この場合は，監視対象ノード $p$ にハートビートメッセージを一定間隔 $\Delta_i$ に送信する実装を導入し，$\Delta_i$ は $p$ と $q$ の間で共有する必要がある．Ping 型の実装は，RFC792 で標準化されている ICMP（Internet Control Message Protocol）を用いて行うため，$p$ に特別な実装を施す必要がなく，また多種のノードを監視することができる．そのうえ，$\Delta_i$ は $q$ の故障検出器のみ知っていればよいため，メッセージの送信間隔は故障検出器が任意に設定することができる（図 2 参照）．

ACCMOS では，より多種のノードを監視することが可能な Ping 型の通信によって監視対象ノードのモニタリングを行う．ACCMOS の実装は，基本的に Java で行ったが，Ping 型の通信に関しては C 言語で ICMP パケットの送受信を実装し，JNI を用いて呼び出すようにした．

### 3.2　データの蓄積と suspicion level $\varphi$ の計算

ACCMOS の実装の概略を**図 3** に示す．

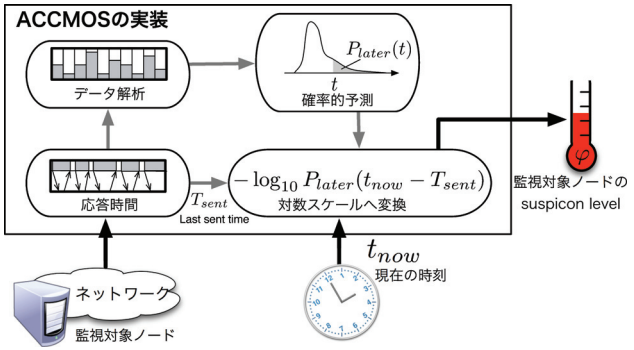



**図 3**　ACCMOS の実装
Fig. 3　The implementation of ACCMOS.



© 2010 Information Processing Society of Japan

まず，ICMP Echo 要求（request）/応答（reply）メッセージによってノード $q$，$p$ 間のラウンドトリップ時間を得る．ACCMOS は，$\varphi_p$ を計算するために，ラウンドトリップ時間を蓄積する．蓄積するデータ $sum$ には次のように重み付けされる．

$$sum := sum \times \omega + RTT_{new} \times (1 - \omega) \quad (2)$$

$\omega$（$0 < \omega < 1$）はすでに蓄積されたラウンドトリップ時間の重みを示し，新たに得たラウンドトリップ時間 $RTT_{new}$ の重みは $1 - \omega$ となる．

ICMP Echo 要求パケットを最後に送信した時刻を $T_{sent}$，現在の時刻を $t_{now}$ としたとき，$t_{now}$ における $p$ が正常である確率[⋆1]は $P_{later}(t_{now} - T_{sent})$ となる（図 3 参照）．したがって，時刻 $T_{sent}$ に $q$ から $p$ へ送信した ICMP Echo 要求パケットに対応する応答パケットが，$q$ においてすでに受信されている場合，$P_{later} = 1$ となる．この値は，蓄積されたラウンドトリップ時間 $sum$ の平均値と累積分布関数（CDF）を用いることによって計算される．ある ICMP Echo 要求/応答パケットによって計測されるラウンドトリップ時間は他の ICMP パケットと独立でかつ ICMP パケットはランダムに遅延すると仮定する．そのため指数分布の累積分布関数を用い，実装は Apache Commons Math version 2.1 [17] を用いて行った．

suspicion level $\varphi_p$ は監視対象ノード $p$ が故障している度合いを示す．したがって，$P_{later}$ が減少するにつれて $\varphi_p$ は増加しなければならない．ただし，ネットワーク越しに監視している故障検出器は $p$ の故障を 100%検出することはできないので，$P_{later}$ は 0 に限りなく近づくが，0 にはならない．$\varphi_p$ はこのような性質を満たすために，以下に示すとおり，$P_{later}$ を対数スケールに変換して算出される．

$$\varphi_p(t_{now}) \stackrel{\text{def}}{=} -\log_{10} P_{later}(t_{now} - T_{sent}) \quad (3)$$

監視対象ノード $p$ が故障している場合，ノード $q$ から送信された ICMP Echo パケットに対応する ICMP Reply パケットは戻ってこないので，$\varphi_p$ は時間の経過とともに $\infty$ に近づく（漸近的完全性，将来的な単調増加性）．一方，$p$ が正常であれば，$q$ において $p$ からの ICMP Reply パケットが受信されるため，$\varphi_p = 0$ となる（上限の存在，リセット）．これにより，$\varphi_p$ は 2.1.2 項で定義した suspicion level の性質を満たすことが分かる．

#### 3.2.1　監視対象ノードの故障判定

ACCMOS は監視対象ノード $p$ の故障判定をユーザやアプリケーション側で個別に行うことによって，ユーザの要求に適応した故障検出サービスの提供とスケーラビリティを両立させている．監視対象ノード $p$ の suspicion level $\varphi_p$ は，ACCMOS に問合せを行うことで得ることができる．ユーザやアプリケーションはそれぞれの要求を反映したしきい値 $\Phi_p$ を持ち，$\varphi_p > \Phi_p$ となったときに故障と判定する．しきい値 $\Phi_p$ は，監視対象ノード $p$ の故障している度合いがどのくらい高まれば故障と判定するかを示す．たとえば，$\Phi_p = 1.0$ ならば，故障している確率が 90%以上（$P_{later} < 0.1$）になれば故障と判定することを意味する．$\Phi_p$ の値を上げれば，故障検出に時間はかかるが，精度の高い故障検出を行うことができる．一方，$\Phi_p$ の値を下げれば，精度は下がるが，故障検出にかかる時間は短くなる．

$\Phi_p$ はそれぞれのユーザやアプリケーションが独自に設定するため，ACCMOS は $\varphi_p$ を提供するだけで，個々の要求した故障検出精度に応じた故障検出サービスを同時に提供することができる．

#### 3.2.2　ACCMOS の利点

ACCMOS は監視対象ノード $p$ の情報を $\varphi_p$ という連続的な値として提供するため，この値を温度計のように表示することで監視対象ノードの状態を分かりやすくすることができる（図 3 参照）．また，ACCMOS は故障判定のために要求する故障検出精度 $\Phi_p$ をパラメータとして与えるため，トラフィックの傾向が変化しても与えられた精度で適切に監視し，ユーザによる設定変更を必要としない．

## 4. 評 価 実 験

### 4.1　故障検出器の評価指標

評価基準は Chen らによって提案された故障検出器の QoS 指標を用いる[3]．本実験では以下の評価指標を用いる．

**Definition1（平均誤検出レート（Mistake Rate）$\lambda_M$）** 故障検出器が単位時間あたりに故障していない監視対象ノードを誤検出する平均回数

**Definition2（正検出確率（Query Accuracy Probability）$P_A$）** 故障検出器が監視ノード $p$ の状態に関する正しい情報を提供する確率

**Definition3（平均検出時間（Average Detection Time）$T_D$）** 監視ノード $p$ が故障していた場合，故障検出器が $p$ に対して故障判定を行うまでの平均時間

$\lambda_M$ は単位時間に誤検出を行う平均回数であるが，誤検出回数が同一でも誤検出している時間[⋆2]が異なる場合は，$\lambda_M$ が同一でも $P_A$ が異なる．これらの指標によって ACCMOS に

⋆1 つまり，$q$ において $p$ からの ICMP Echo 応答パケットが $t_{now}$ 以降に到着する確率．

⋆2 図 2 において $p$ は正常であるが，$q$ の出力値が *suspect* となっている時間．



© 2010 Information Processing Society of Japan

おける故障判定の偽陽性（false positive）を評価する．故障判定の偽陽性は分散合意アルゴリズムの性能低下を招くなどの弊害があり[15]，$\lambda_M$ や $P_A$ は故障検出器の性能評価指標として広く用いられている．アクルーアル故障検出器である ACCMOS は設定可能なパラメータとしてのタイムアウトは持たないが，あるしきい値 $\Phi_p$ を設定すると，監視対象を故障と判定するために，ネットワークの状態に適応した暗黙のタイムアウトが存在する．この実験において $T_D$ は，この暗黙のタイムアウトがどのように設定されているかということを $\Phi_p$ の変化とともに観測するための指標である．

### 4.2　実験環境と ACCMOS の設定

実験は以下の 3 つの環境で行った．

**実験環境 1（Env.1）：** IEEE802.11n の無線ネットワーク．Apple 製 Time Capsule を用いて無線 LAN を構成し，監視元ノードは同一サブネット内の監視対象ノードを監視した．

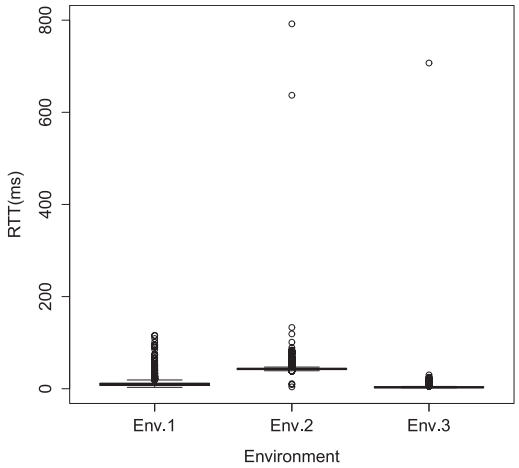
**実験環境 2（Env.2）：** 国内の広域ネットワーク（WAN）．監視元ノードを京都市内自宅に設置し，WAN を介して京都産業大学内（同市内）に設置した監視対象ノードを監視した．自宅から ISP までは 8 Mbps の ADSL で接続されており，学内 LAN は 100 Mbps である．

**実験環境 3（Env.3）：** 家庭内 PLC（Power Line Communications）ネットワーク．同一家屋内において，パナソニック製 PLC アダプタ BL-PA510 を 3 台使用して有線ネットワークを構成した．監視対象ノードと監視元ノードはそれぞれ異なる PLC アダプタのスレーブに接続されている．監視元，監視対象ともに同一サブネット内に設置されている．

監視対象ノード $p$ と ACCMOS が動作している監視元ノード $q$ はともにつねに正常に動作しており，ACCMOS が動作しているノード $q$ から 3 秒（3,000 ミリ秒（ms））ごとに ICMP Echo 要求を送信し，$p$ から Echo 応答を受け取った際にラウンドトリップ時間（RTT）を計測している．実験の結果としては，$q$ において動作している ACCMOS の平均誤検出レート $\lambda_M$ と正検出確率 $P_A$ を得ることができる．

パケットロスが起きた際の RTT は正確には $\infty$ 秒であるが，今回の実装においては 2500 ms として扱った．実験に用いた環境において RTT の最大値は Env.1 で 116 ms，Env.2 で 792 ms，Env.3 で 707 ms となっており，2,500 ms 以上経過して届く ICMP Echo 応答はいずれの実験環境においてもなかったため，RTT の十分大きな値としてこれを用いた．

実験を行った環境における RTT の分布は**図 4** のとおりである．ただし，パケットロスとして扱った RTT は集計せず，パケットロス率として表示している．図 4 から分かるとおり，RTT の分布が最も安定していたのが Env.1 であった．一方，Env.3 は図 4 のグラフを見ると安定しているように見えるが，パケットロスが非常に多く，バースト状態も起きていた．Env.2 は接続環境が WAN であるため，RTT の分布は Env.1 や Env.2 より少しばらついているが，実験環境の中ではパケットロス率が最も低く，パケットロスに影響されない実験結果を得ることができた．




| | ラウンドトリップ時間 (RTT) | | | |
|---|---|---|---|---|
| 実験環境 | 中央値 (ms) | 平均値 (ms) | 標準偏差 | パケットロス率 (%) |
| Env.1 | 10.00 | 9.77 | 4.40 | 0.43 |
| Env.2 | 43.00 | 43.50 | 9.64 | 0.11 |
| Env.3 | 3.00 | 3.62 | 6.07 | 2.30 |

**図 4**　各実験環境におけるラウンドトリップ時間（RTT）の分布とパケットロス
Fig. 4　On round trip time (RTT) and packet loss in each environment.

ACCMOS のパラメータとしては，故障判定のしきい値を，$\Phi_p \in \{0.7, 1.0, 2.0, 3.0\}$（それぞれ，図 3 における $P_{later} = \{0.2, 0.1, 0.01, 0.001\}$ に対応）とした．これらのパラメータについて，それぞれの実験環境で 3 時間実験を行った．なお，ACCMOS が $\varphi_p$ を計算する際に用いる重みは $\omega = 0.8$ としている．

### 4.3　実 験 結 果

前述のネットワーク環境において実験を行い，実験結果として**表 1** が得られた．平均誤検出レート $\lambda_M$ は，ACCMOS が 1 秒間に起こす誤検出の平均回数を示している．**図 5** はその


© 2010 Information Processing Society of Japan

**表 1**　実験結果：平均誤検出レート $\lambda_M$ と正検出確率 $P_A$
Table 1　Summary of the experimental result.

| 指標 | 実験環境 | $\Phi_p = 0.7$ | $\Phi_p = 1.0$ | $\Phi_p = 2.0$ | $\Phi_p = 3.0$ |
|---|---|---|---|---|---|
| $\lambda_M$ | Env.1 | 0.0412 | 0.0022 | 0.0013 | 0.0007 |
| | Env.2 | 0.0014 | 0.0004 | 0.0004 | 0.0004 |
| | Env.3 | 0.0513 | 0.0082 | 0.0060 | 0.0054 |
| $P_A$ (%) | Env.1 | 86.7 | 99.4 | 99.6 | 99.8 |
| | Env.2 | 99.6 | 99.8 | 99.8 | 99.9 |
| | Env.3 | 84.6 | 97.8 | 98.8 | 99.4 |

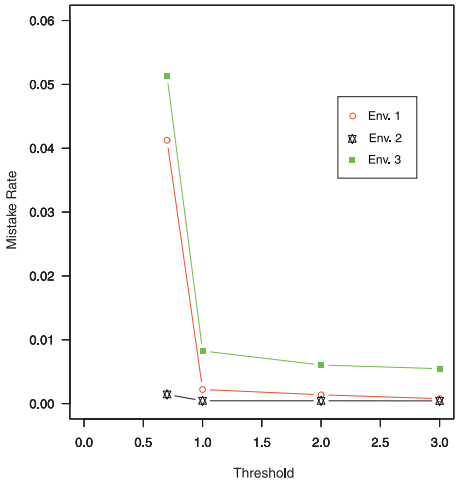



**図 5**　平均誤検出レート（Mistake Rate）$\lambda_M$ としきい値（Threshold）$\Phi_p$
Fig. 5　Mistake rate $\lambda_M$ and threshold $\Phi_p$.

$\lambda_M$ をプロットしたものだが，$\Phi_p \geq 1.0$ のときにはすべての実験環境において $\lambda_M \leq 0.01$ となっており非常に誤検出率が低い結果となった．

ユーザ側から見たとき，要求する故障検出精度をしきい値 $\Phi_p$ として与えるため，実際に得られた故障検出精度である $P_A$ は非常に重要な指標である．$\Phi_p = \{0.7, 1.0, 2.0, 3.0\}$ はそれぞれ 80%，90%，99%，99.9%の故障検出精度に対応する．特に Env.2 のようなパケットロスがほとんど起こらない環境では，きわめて高い故障検出精度を示した．Env.1 では，$\Phi_p = 3.0$ のとき $-0.1$ ポイント，Env.3 では，$\Phi_p = 2.0$ のとき $-0.2$ ポイント，$\Phi_p = 3.0$ のとき $-0.5$ ポイントと若干下回っているものの，全体的には，ほぼ要求する故障検出精度

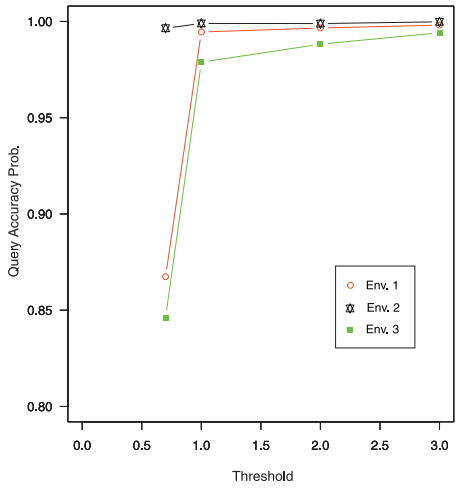



**図 6**　正検出率（Query Accuracy Prob.）$P_A$ としきい値（Threshold）$\Phi_p$
Fig. 6　Query accuracy probability $P_A$ and threshold $\Phi_p$.

と同等かそれ以上の故障検出精度が得られた（表 1，**図 6** 参照）．

**図 7** は各しきい値 $\Phi_p$ における $P_A$ と $T_D$ の変化についてプロットしたものである．Env.3 は図 5，6 において Env.1 と似通ったカーブを描いているが，$T_D$ に関しては大幅に遅くなっている．これは，パケットロスを通常の RTT と同様に扱っているためであるが，Env.1 と比較すると，Env.3 は 5 倍以上パケットロスが起こっており，$\Phi_p$ を上げて高精度の故障検出を要求する場合，これらが単なる外れ値ではなく，性能に大きく影響することを示している．図 7 の $T_D$ はミリ秒オーダなので，今回の実験では体感するほどの性能低下ではないが，パケットロスの取扱い方によって故障検出時間が意図せず悪化する可能性があるので，この点は再考する必要がある．

有線 LAN における $\varphi$ 故障検出器の実験結果は，$\Phi_p = 1$ のとき $\lambda_M = 0.0897$，$\Phi_p = 2$ のとき $\lambda_M = 0.0249$，$\Phi_p = 3$ のとき $\lambda_M = 0.0063$ であった[7]．パケットロス率に関していえば，$\varphi$ 故障検出器の実験環境は 0.05%であり，今回の実験環境は Env.1 で 0.43%，Env.2 で 0.11%，Env.3 で 2.30%であり，ACCMOS はパケットロス率がより高い環境で実験を行ったことになる．したがって，ACCMOS は偽陽性に関して $\varphi$ 故障検出器より性能が大幅に向上しているといえる（表 1，図 5）．


© 2010 Information Processing Society of Japan

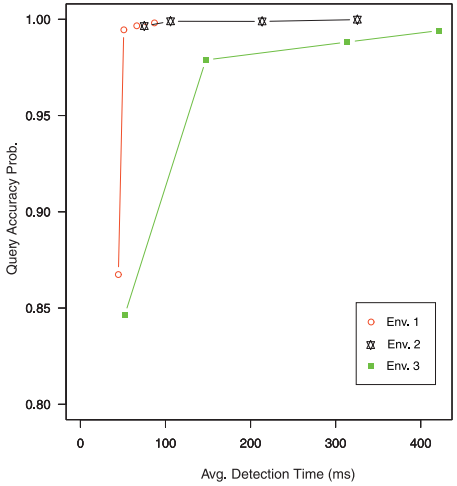



**図 7**　平均検出時間（Avg. Detection TIme）$T_D$ と正検出率（Query Accuracy Prob.）$P_A$
Fig. 7　Average detection time $T_D$ and query accuracy probability $P_A$.

## 5. まとめと展望

本論文では，アクルーアル故障検出器 ACCMOS の設計と実装について述べ，複数のネットワーク環境における実験により性能評価を行った．

評価実験の結果，ACCMOS は様々な環境において高精度の故障検出を実現することが確認された．また，同様の環境において，$\varphi$ 故障検出器よりも監視対象ノードの誤検出を大幅に低減することも確認された．

今回の ACCMOS の実装ではパケットロスに関して，十分大きな RTT（2,500 ms）を割り当てたが，今後はネットワーク環境に応じて，正常な範囲のパケットロスなどを適切に処理するすることによって故障検出精度 $P_A$ の向上や平均検出時間 $T_D$ の増加を抑制する必要がある．

またモニタリングシステムとしては，ACCMOS に加えて，手軽にホームネットワーク環境内のノードを監視できるようにするために，近傍ノードを探索し，それらの監視を開始する初期設定自動化機構と ACCMOS の出力を分かりやすく表示するための GUI を実装する．

**謝辞**　本研究は科研費（No.20700072），および京都産業大学総合研究支援制度（No.530）の助成を受けたものである．

## 参 考 文 献

1) Bertier, M., Marin, O. and Sens, P.: Implementation and performance evaluation of an adaptable failure detector, *Proc. 15th Int'l Conf. on Dependable Systems and Networks* (*DSN'02*), Washington, D.C., USA, pp.354–363 (2002).
2) Chandra, T.D. and Toueg, S.: Unreliable failure detectors for reliable distributed systems, *J. ACM*, Vol.43, No.2, pp.225–267 (1996).
3) Chen, W., Toueg, S. and Aguilera, M.K.: On the Quality of Service of Failure Detectors, *IEEE Trans. Comput.*, Vol.51, No.5, pp.561–580 (2002).
4) Défago, X., Urbán, P., Hayashibara, N. and Katayama, T.: Definition and Specification of Accrual Failure Detectors, *Proc. Int'l Conf. on Dependable Systems and Networks* (*DSN'05*), Yokohama, Japan, pp.206–215 (2005).
5) DIALNP: APPIA Communication Framework. http://appia.di.fc.ul.pt
6) Facebook: Facebook. http://www.facebook.com
7) Hayashibara, N. and Takizawa, M.: Performance Evaluation of the $\varphi$ Failure Detector, *Proc. 8th IEEE ICDCS Workshops* (*MNSA'06*) (2006).
8) Hayashibara, N., Défago, X., Yared, R. and Katayama, T.: The $\varphi$ Accrual Failure Detector, *Proc. 23rd IEEE Int'l Symp. on Reliable Distributed Systems* (*SRDS'04*), Florianópolis, Brazil, pp.66–78 (2004).
9) Jacobson, V.: Congestion Avoidance and Control, *Proc. ACM SIGCOMM'88*, Stanford, CA, USA (1988).
10) Lima, R., Cirne, W., Brasileiro, F., Fireman, D. and Distribudos, L.D.S.: A case for event-driven distributed objects, *Proc. DOA'06*, LNCS 4276, pp.1705–1721, Springer-Verlag (2006).
11) Nagios Enterprises: Nagios. http://www.nagios.org
12) NTT DATA CORPORATION: Hinemos. http://www.hinemos.info
13) OurGrid: The OurGrid Community Web Site. http://www.ourgrid.org
14) Project, X.: Xymon. http://www.xymon.com
15) Sampaio, L.M.R., Brasileiro, F.V., Cirne, W. and Figueiredo, J.C.A.: How Bad Are Wrong Suspicions? Towards Adaptive Distributed Protocols, *Proc. IEEE Intl. Conf. on Dependable Systems and Networks* (*DSN'03*), pp.551–560 (2003).
16) The Apache Cassandra Project: Cassandra. http://cassandra.apache.org
17) The Apache Software Foundation: Apache Commons Math. http://commons.apache.org/math/
18) Twitter: Twitter. http://twitter.com

(平成 22 年 4 月 6 日受付)
(平成 22 年 9 月 17 日採録)



© 2010 Information Processing Society of Japan

## 推　薦　文

本論文は，システム管理のための高度なモニタリングシステムの実現を目的に，アクルーアル型故障検出器 ACCMOS を提案している．この論文の提案手法によって，監視対象ノードの suspicion level を連続的な値が提供可能になり，その結果，ユーザビリティが高く精度の高いシステムを実現することになる．さらにこの論文では，無線ネットワーク環境において性能評価を行い，その有効性を検証している．以上のことから読者にとって十分に価値の高い論文であることが認められ推薦論文として掲載されることとなった．

（マルチメディア通信と分散処理研究会主査　串田高幸）

**林原　尚浩**（正会員）

1976 年生．1999 年和歌山大学経済学部社会システム設計学科卒業．2001 年北陸先端科学技術大学院大学情報科学研究科博士前期課程修了．2004 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．北陸先端科学技術大学院大学情報科学研究科研究員を経て，2005 年より東京電機大学理工学部情報システム工学科助手．2008 年より京都産業大学コンピュータ理工学部助教．現在に至る．耐故障分散システム・アルゴリズムの研究に従事．IEEE，ACM 各会員．



© 2010 Information Processing Society of Japan