社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－134 (43)
2008－CSEC－ 40 (43)
2008／3／7

リッチメディアコミュニケーションにおける障害検出方式の提案

中島一彰†、金友大、大芝崇、子林秀明
NEC サービスプラットフォーム研究所
†nakajima@ah.jp.nec.com

概要

音声、映像、情報共有ツールを同時に利用するリッチメディアのコミュニケーションにおいて、端末やサーバを過負荷にすることなくコミュニケーションの障害を高速に検出する方式を提案する。リッチメディアコミュニケーションではすべてのツールが正常に動作しなければならず、そのためにすべてのツールで発生する障害を即座に検出して、コミュニケーションの中断を最小限に抑えることが求められている。しかし、複数のツールが独立して障害監視を行うと、端末やサーバの CPU 負荷が上昇してしまい、音切れや映像の乱れなど、コミュニケーションの品質が劣化する。そこで、ツールごとの障害監視の頻度を、CPU 負荷とツールの優先度に応じてミドルウェアで統合的に制御することで、障害の検出時間を最小化する方式を提案する。

A Proposal of the Fault Detection Method for Rich Media Communications

Kazuaki Nakajima, Dai Kanetomo, Takashi Oshiba and Hideaki Nebayashi
NEC Corporation, Service Platforms Research Laboratories

ABSTRACT

We propose a fast fault detection method for rich media communications using voice, video and information sharing tools without the occurrence of overload on servers and terminals. It is necessary to minimize interruption of communications by detecting any trouble on all tools. Quickness of the detection depends on intervals of health check on the tools, but high-frequent health check causes quality deterioration of communications by overloading servers and terminals. We propose the method of minimizing fault detection time by dynamically controlling the health check interval on each tool according to CPU load and priority of the tools with middleware.

1. はじめに

近年、正確な情報伝達に基づく意思決定の迅速化への期待から、電話、Web 会議、メール、チャットなどのコミュニケーション手段を統合したユニファイドコミュニケーション（以下、UC）が注目されている[1]。UC により、今までのような電話による音声通話だけでなく、音声、映像の通話ツールに加えて、AP 共有や Web 共有、資料共有などの情報共有ツールを同時に利用したリッチメディアコミュニケーションが可能になる。

しかし、リッチメディアコミュニケーションでは、いずれかのツールで障害が発生すると正確なコミュニケーションができなくなる。例えば、音声だけは通じていても、同時に利用している AP 共有で障害が発生すると、話している内容と相手が見ている画面とがずれてしまい、正しい情報を伝えることができなくなる。

したがって、ツールの障害を瞬時に検出して、ツールの障害を復旧する必要がある。ところが、障害を瞬時に検出するために、障害監視の頻度を上げると、端末およびサーバの装置の CPU

が過負荷になってしまう。CPU が過負荷になると、コミュニケーションの処理に遅延が発生し、音切れや映像の乱れ、情報共有ツールの同期ずれなどが生じ、コミュニケーションの品質が劣化する。

本稿では、音声、映像、情報共有ツールを同時に利用するリッチメディアコミュニケーションにおいて、CPU 負荷と各ツールの優先度に応じて統合的に障害監視を行う障害検出方式を提案する。

2. 障害検出の要件と従来技術の課題

2.1. リッチメディアコミュニケーションにおける障害検出の要件

リッチメディアコミュニケーションには、ツールが個別に高頻度な障害監視をすると CPU 負荷が上昇すること、利用者が注目するツールは限られるという 2 つの特性がある。そこで、リッチメディアコミュニケーションでの障害検出の要件は次の 2 点となる。

(1) CPU が過負荷になることを防止

障害確認の頻度が高くなることの影響で CPU が過負荷になり、コミュニケーション品質が劣化することを防止する必要がある。

(2) 注目したツールの障害検出を優先

複数のツールを同時に利用するときに、主に使うツールを優先して障害検出に要する時間を短縮する必要がある。

リッチメディアコミュニケーションで高頻度な障害監視を個別のツールが独立して実施すると、端末やサーバなどの装置の CPU 負荷が上昇し、処理能力の限界を越え、コミュニケーションの遅延や揺らぎが発生し、品質が劣化する。例えば、音声においては、途切れなくパケットが相手に届いているかを高頻度で監視すると、データの照合、ヘルスパケットの送受信、障害監視に必要なタイムアウトスレッドの動作などで CPU に負荷がかり、遅延や音切れが発生する。したがって、多くのシステムでは正常動作の安定性を優先して各ツールの障害監視の頻度を低くしており、障害が発生したときの障害検出に時間がかかる。

また、リッチメディアコミュニケーションでは、利用者は常に全ツールを注目しているのではなく、状況に応じて注目するツールを変えている。例えば、映像と資料共有ツールを同時に使っている場合に、相手の表情を確認したいときには映像に注目し、資料を説明しているときには資料共有ツールに注目する。注目しているツールで障害が発生するとコミュニケーションが不可能になるので、注目しているツールの障害検出を優先する必要がある。

2.2. 従来技術の課題

障害検出の従来技術はデータ解析と死活監視の 2 つがある。

音声品質を定期的に確認する RTCP XR を用いた方式[2]に代表されるデータ解析の方式では、データ解析に要する CPU 処理量が大きい。CPU 負荷は品質確認の頻度と対象の数の積に比例するので、品質確認を高頻度に多数の機器を対象に行うと、CPU が過負荷になりやすい。例えば、会議サーバなど多数の端末を収容する装置では、頻度を高く保ったまま品質確認を実行すると、会議サーバの CPU が過負荷になる。

定期的にヘルスパケットを送受信する死活監視の方式は、ネットワークレベルでの障害検出方式として、ICMP の Echo パケットを利用した Ping 方式がある[3][4]。また、AP レベルでは通信相手のハートビートを AP レベルのヘルスパケットで確認する方式がある。これらの方式は障害監視の頻度を一定に保ち、一定間隔に保たれるべきヘルスパケットの送受信において異常値があるかを分析する。したがって、ヘルスパケットの送信間隔の動的な間隔の変更に弱く、ツールの優先度に応じて障害検出の頻度を動的に変更することができない。

また、障害監視の頻度を調整する死活監視の

方式として、ネットワークの負荷を考慮した方式があり[5]、通信の流量に基づいてヘルスチェックの間隔を調整する。しかし、この方式はネットワーク負荷に基づいた一様な調整しかできないので、障害監視の頻度をツールごとに設定することができない。

3. 提案する統合型障害検出方式

3.1. 障害検出システムの設計方針

リッチメディアコミュニケーションにおける障害検出の2つの要件を満たすには、ミドルウェアで統合的に障害検出を制御する必要がある。2つの要件を満たす障害検出システムの設計方針として次の2つを立てた。

(1) CPU 負荷に基づいたフィードバック制御

ミドルウェアにおいて、障害監視の頻度の変更を CPU 負荷の測定に基づいて実施することで、端末装置やサーバ装置の CPU が過負荷になることを防止する。

(2) 優先度に基づいた障害監視の頻度変更

ツールごとに優先度を設定し、優先度に基づく障害監視の頻度の変更をミドルウェアで行うことで、優先度が高いツールの障害監視の頻度を優先的に高くする。

3.2. モジュール構成

図 1 にモジュール構成を示す。障害検出ミドルウェアが各ツールの障害検出エンジンを制御する。CPU 負荷の測定と各ツールの優先度変更を可能にすることで、CPU 負荷とツールの利用状況に応じた、障害監視の頻度の変更を実現することができる。

次に、**図 2** に想定するシステム構成を示す。多人数の電子会議を可能にする構成を想定しており、複数の端末装置がサーバ装置に接続するシステム構成となっている。端末装置、サーバ装置それぞれに障害検出ミドルウェアが動作する。障害検出ミドルウェアには各ツールの障害監視の頻度を高めるとともに、各装置の CPU の過負荷防止を実現する。

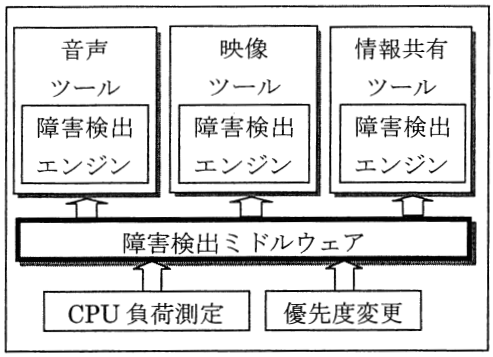


図 1 モジュール構成

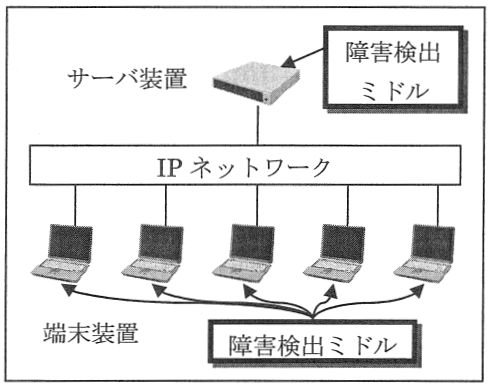


図 2 システム構成

3.3. 統合型障害検出方式の制御モデル

図 3 に統合型障害検出方式の制御フローを示す。本制御モデルの特徴は、CPU 負荷とツールごとの特性に応じて、各ツールの障害監視の頻度を決定することにある。

まず、コミュニケーションの実行前に、ツールごとに障害監視の最低限度の頻度と、最高限度の頻度を設定する。

次に、コミュニケーションの実行中に障害検出を行うプロセスを実行する。利用場面に応じてツールごとの優先度の変更を動的に行う。そして、定期的に障害監視の頻度の変更を行うために、CPU 負荷を測定し、CPU 負荷が目標値を越えているかを判断する。さらに、装置全体の障害監視の頻度調整の進行度を表す「進行度変数」を用いて、CPU 負荷に適した障害監視の頻度を算出する。CPU 負荷が目標値内であ

ると進行度変数を減少させ、CPU 負荷が目標値を超えると、進行度変数を増加させる。次に、進行度変数と優先度に基づいて、3.4.4 で述べる軽量な算出方式によって、障害監視の頻度をツールごとに算出する。進行度変数の変更の結果、進行度変数が減少すると優先度に基づいて各ツールの障害監視頻度を大きくし、進行度変数が増加すると障害監視頻度を小さくする。最後に、障害監視を実施する。

一定時間経過後に優先度の変更と進行度変数の再計算を繰り返し、コミュニケーション実行中に障害監視の頻度の動的な変更を行う。

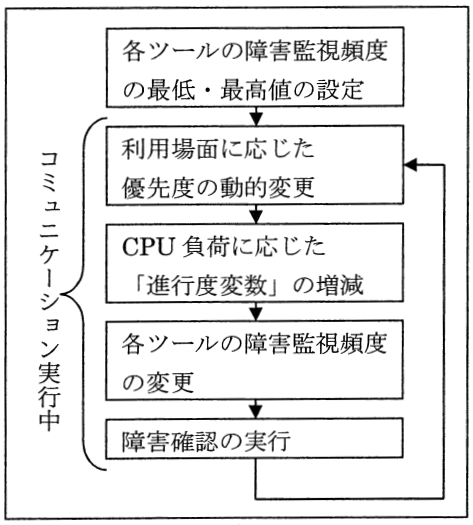


図 3 統合型障害検出方式の制御フロー

3.4. 統合型障害検出方式の動作

3.4.1. 各ツールの障害監視頻度の最低・最高値の設定

音声、映像、および情報共有ツールごとに障害監視の最低頻度と最高頻度を設定する。

障害検出の基本動作は、障害検出対象のツールから障害検出管理側である障害検出ミドルウェアにヘルスパケットを送信し、正しいヘルスパケットを一定間隔で受信しているかどうかで判定する。障害監視の頻度は、ツールの特性によって設定が必要である。

帯域が一定しているストリーム型の通信を行う音声と映像のツールでは利用する帯域に応じて設定する。音声はデータ量が小さく、音切れなどの障害は発話者に瞬時に知らせる必要があるので、障害監視の頻度の最低・最高値は高く設定する。また、映像はデータ量が大きく、通信の遅延のばらつきが大きくなるので、バッファリングの許容範囲が広く、障害監視の頻度の最低・最高値は音声よりも低く設定する。

また、間欠型の通信が行われる資料共有やチャットなどの情報共有ツールは帯域が一定しないので、想定される最大のデータ量に応じて、障害監視の頻度の最低・最高値を設定する。

3.4.2. 利用場面に応じた優先度の動的変更

障害検出のツールの優先度を動的に変更し、高優先のツールの傷害検出の頻度を優先的に高くする。一般的なコミュニケーションでは音声は常に優先度を最高に設定する。一方、映像と情報共有ツールは利用場面に応じて、優先度を変更する。資料共有などの情報共有ツールを操作している場合には、情報共有ツールの優先度を向上し、映像を最大化して、相手の表情を確認している場面では、映像の優先度を向上する制御を行う。

3.4.3. CPU 負荷に応じた進行度変数の増減

CPU 使用率に基づいた CPU 負荷の測定を行い、進行度変数の増減を行う。CPU 使用率の実測値を用いることで、ほかのプロセスの影響による CPU 使用率の増減に対応する。CPU 負荷の目標値を設定し、現在の CPU 負荷が目標値以下であったら、進行度変数を減算し、現在の CPU 負荷が目標値以上になったら、上限まで順に進行度変数を加算する。

3.4.4. 各ツールの障害監視頻度の変更

進行度変数に基づき、優先度が低いツールから段階的に障害監視の頻度を下げる制御を行う。本方式では、同時に利用するコミュニケーションの数を 5 つまで対応できるように、優先度を 5 段階とした。優先度 1 が最高優先度、優

先度5が最低優先度である。

そして、進行度変数に応じて、6段階の頻度等級を算出し、頻度等級から各ツールの障害監視の頻度を決定する。進行度変数は0から25の値を取る。頻度等級の変更に優先度を反映させるために、優先度ごとに段階的な頻度等級を決定する次の方式を用いる。優先度ごとに進行度変数に基づいた頻度等級が指定される。表1に優先度ごとに進行度変数の値で決定される頻度等級を示す。各ツールは優先度にしたがって表内に示した進行度変数の現在の値を超えない範囲でもっとも小さい頻度等級を選択する。障害監視の頻度は次の数式で算出する。

頻度＝最低頻度+(最高頻度－最低頻度)
×頻度等級／5

表1 進行度に基づいた頻度等級の決定

頻度等級	優先度（高→低）				
	1	2	3	4	5
0	25	23	20	16	11
1	24	21	17	12	7
2	22	18	13	8	4
3	19	14	9	5	2
4	15	10	6	3	1
5	上記より小さい場合				

（表内の数値は進行度変数の値）

4. 方式検証と考察

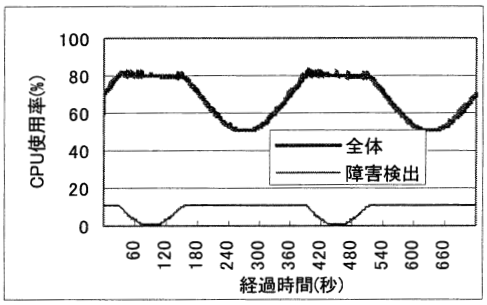
4.1. 方式検証におけるシステム構成

サーバに複数の端末が接続する環境において[6]、本障害検出の方式検証を行った。1つのサーバに20～60端末が接続し、各端末で5つのテストツールが動作するシミュレートシステムで、サーバの障害監視の頻度を測定した。サーバのCPU使用率の目標値は、サーバの能力を最大限発揮し、かつ、サーバが過負荷に陥る危険が少ない80%とした。

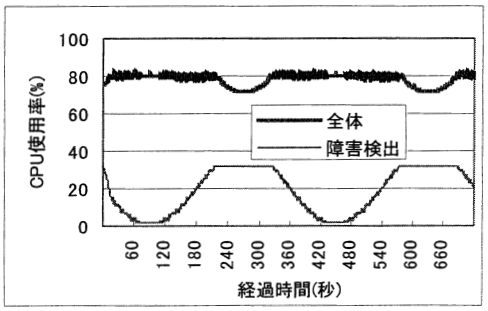
4.2. CPU負荷と障害監視の頻度の変化

周期360秒で0%から40%のCPU負荷を与えるCPU負荷発生テストAPで、サーバに負荷を与えた場合をシミュレートし、CPU使用率を算出した。各ツールの障害監視の頻度は、最低で1分に2回、最高で1分に60回に設定した。また、テストツールには、5つの優先度を均等に割り当てた。今回の検証では、20端末100ツールを対象としたパタンAと、60端末300ツールを対象としたパタンBで行った。

その結果、図4に示すように、パタンA、パタンBともにCPU負荷発生テストAPが与えるCPU負荷を打ち消すように、障害監視のCPU使用率が低下し、全体のCPU使用率が80%を大きく越えないことが分かった。パタンAでは90秒経過時点で最高で90%まで負荷が上昇するはずだったが、障害検出の負荷分の10%を削減して約80%に抑えていた。また、パタンBのように管理対象のツール数が多くなると、全体のCPU負荷が上がることで、障害監視の頻度調整が広い期間で実行された。



パタンA 管理対象が100ツールの場合



パタンB 管理対象が300ツールの場合

図4 CPU使用率の変化

次に、優先度ごとの障害監視の頻度の変化を検証する。CPU 負荷が上昇する場合での、優先度別の障害監視の頻度の変化を図 5 に示す。CPU 負荷が軽く CPU 使用率が 80%以下のときには、どの優先度も高頻度で障害監視を行っている。そして、全体の CPU 負荷が上がり、障害検出の CPU 使用率が低下するにつれて、低優先度の障害監視から頻度が低下する。

例えば、経過時間 10 秒での各ツールの障害検出の頻度は、優先度 1 のツールでは最高頻度である 1 秒ごと、優先度 3 のツールでは 1.6 秒ごと、優先度 5 のツールでは 3.8 秒ごとになった。優先度が高いツールでは高頻度に障害監視が実行され、優先度が低いツールになるにつれて頻度が低くなることが証明された。

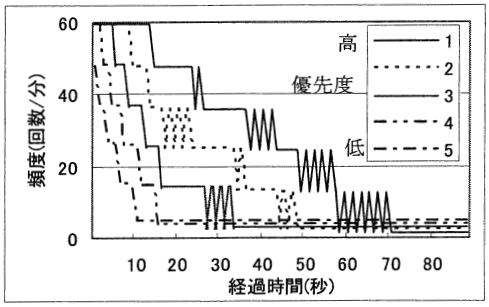


図 5 優先度別の障害監視頻度の変化

4.3. 考察

方式検証の結果、サーバの CPU が過負荷になることを防止し、障害監視の頻度がツールの優先度に応じて段階的に低下していくことがわかった。これは、CPU の過負荷の影響でコミュニケーションの品質が劣化しない範囲で、各ツールの障害監視の頻度が高く設定されたことを示している。

しかし、さらに端末数が多くなり、同時に監視対象のツール数が多くなると、各ツールの障害監視の頻度を最低にしても、CPU 使用率が目標値を越える危険がある。この問題を解決するには、メディアの品質を落とすなどの、適応的なリソース割当制御[7]と組み合わせる必要がある。また、利用者の増加に伴う監視対象のツール数の増加に対応できるように、優先度を 5 段階からさらに細分化して、利用者ごとに細かく優先度を割り当てられるようにすることが有効である。

5. おわりに

本稿は、リッチメディアコミュニケーションにおいてツールの優先度と CPU 負荷に応じて、障害監視の頻度をミドルウェアで統合的に制御する障害検出方式について提案した。今後は、実用化に向けて、実際の音声・映像・情報共有ツールを利用した実証実験を行う予定である。

参考文献

[1] Gartner, Inc., “Magic Quadrant for Unified Communications, 2006”, June 2006

[2] 増田、古川、林、馬島：RTCP XR を用いた End-to-End 音声品質管理方式、信学技法、TM2006-69、pp.47-52、2007.3

[3] 宮内、高野、寺島、ほか：ネットワーク障害診断ツールの開発、信学技法 TM2004-106、 pp.25-30、2005.3

[4] 上野、今井、中後：大規模マルチキャスト配信サービス向け障害監視方式の検討、信学技法 TM2006-30、 pp.53-58、 2006.7

[5] 澤田、中山、花岡：ネットワークシステムへ与える負荷を考慮したヘルスチェック方式の一提案、電子情報通信学会総合大会講演論文集 No2 p98、2003.3

[6] 中島、大芝、金友、田淵：グループ通信における拠点間通信を集約する動的経路制御方式、情報処理学会第 130 回 DPS 研究会、pp.1-6、2007.3

[7] 金友、中島、大芝、田淵：リッチメディアコミュニケーションにおけるリソースの傾斜配分による品質制御方式の提案、情報処理学会 DICOMO シンポジウム、2007.7