# C/S型オンラインゲームの自己適応化に向けたMAPEループ構成パターンの提案

山縣　慧[1]　中川　博之[1]　清　雄一[1]　田原　康之[1]　大須賀　昭彦[1]

**概要**：近年，インターネットの発展に伴い，オンラインゲームはますます人気が高まってきている．しかしオンラインゲームは高いリアルタイム性・相互作用が求められており，さらにゲームを管理するサーバや世界中に分散するユーザの環境の違いを考慮しなければならず，高いパフォーマンスを保つことが難しい．そこで本研究では，環境の変化に応じて，自ら構成や振る舞いを変化させることのできる自己適応システムのアプローチの 1 つとして研究されている MAPE（Monitor(監視)，Analyze(分析)，Plan（計画），Excute(実行)）ループを用いる．C/S 型オンラインゲームシステムの自己適応化に向けて，クライアントの通信環境を考慮し，サーバがオンラインゲームの振る舞いを変化することのできる，MAPE ループ構成パターンの提案・評価を行った．

## 1. はじめに

近年，インターネットの発展に伴い，コンピュータネットワークの利用者は増加し，それに伴いネットワークサービスもますます発展してきている．特にオンラインゲームは世界中に人気がある娯楽サービスであり，多くの利用ユーザが存在する．オンラインゲームは高いリアルタイム性・相互作用が求められており，さらにゲームを管理するサーバや世界中に分散するユーザの環境の違いを考慮しなければならず，高いパフォーマンスを保つことが難しい．多人数のユーザが同じゲーム進行を共有するため，ユーザの行動・ステータス情報を同時に管理しなければならず，また世界中のユーザがサービスをいつでも利用できるよう，サーバには高い可用性が求められる．

また，近年のソフトウェアは，その大規模・複雑化に伴い，環境変化に対しても，人が介在することなく，動的に環境に適応する能力が求められるようになってきている．特に，システムの部分的な故障や突発的な負荷の増加への対処や，実行時の環境の変化に追従したサービス提供が強く求められている．こうしたソフトウェアの複雑さ，環境変化に応じて，自ら構成や振る舞いを変化させることのできる自己適応システム (Self-adaptive Systems) は，QoS(Quality of Service) が要求されるネットワークシステムに対しても有効なアプローチとして研究されており，実現に対する期待が高まっている [1],[2]． 自己適応システムの振る舞いを決定するアプローチとして，MAPE(Monitor, Analyze, Plan, Execute) ループ [3] が挙げられる．MAPE ループをシステム内のサブシステムとして，これらのコンポーネントを適切に分散して配置することは，分散したシステムの相互作用を考慮することができるという観点から有効とされている．しかし，それらの設計方法やドメイン特性の考慮，実際の適用例が少なく，実際のシステムに組み込むにはまだまだ課題が多い．

本論文では，オンラインゲームのようなネットワーク上において高度な相互作用性が必要なシステムが自己適応できる，分散システムにおける新しい MAPE ループ構成パターンを提案する．また，提案パターンを適用した簡易システムを構築・評価することにより，提案パターンがオンラインゲームに適した自己適応が可能であることを示す．

本研究アプローチは，MAPE ループのコンポーネントを分散して配置することで，分散環境における自己適応を実現するため，災害時や過酷な環境下での人の介入できない場所におけるロボット協調動作，複数の離れた場所から，数人が同時に協調して遠隔ロボットによる手術，国際会議におけるテレビ会議，遠隔地同士のオンライン音楽セッション，オンラインゲームなど，非常に高度な相互作用が必要な状況において有効性が期待できる．

本論文は以下のような構成になっている．2 章で自己適応システムの動向と取り組むべき課題点，3 章ではオンライゲームにおける課題点を述べる．4 章では課題解決のコンセプトとモチベーションについて述べ，提案 MAPE ルー

[1] 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, University of Electro-Communications

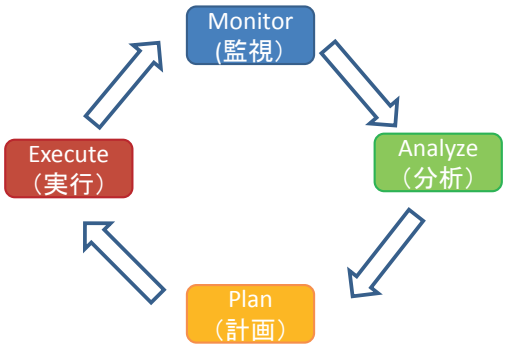



図 1　MAPE ループ

プ構成パターンの概要について説明する．5 章では実行時の適応シナリオでの評価実験を通して有効性があることを示し，考察を述べる．最後に 6 章で，まとめと今後の課題について述べる．

## 2. MAPE ループ

### 2.1 MAPE ループとは

自己適応システムはシステムが自身の機能と目標を管理し，環境変化に応じて柔軟な振る舞いをする能力を持つシステムのことである．自己適応システムは自らの振る舞いを決定するためのアプローチが必要であり，その 1 つとして，MAPE ループと呼ばれるコントロールループが有効であるとされている [3]，[4]，[5]．MAPE ループでは自己適応の主要な機能の要素として，Monitor(監視)，Analyze(分析)，Plan(計画)，Execute(実行) の 4 つのコンポーネントによって構成されている（図 1）． Monitor(監視) ではセンサーを通じてコントロールできないソフトウェア，ハードウェア，ネットワークからシステム環境の情報を集める．Monitor（監視）がシステムの機能の異常を検知したとき，システムは自己適応を行われる．何らかのシステムの機能に異常が Monitor(監視) によって検知された場合，Analyze（分析）は Monitor（監視）で得られた情報を元に，システムが異常をきたしている原因を分析する．Plan（計画）では，Analyze（分析）結果をもとに，システムの構成・振る舞い変更の計画を立て，Execute(実行) では Plan（計画）の情報を元に，システムの機能の構成を組み替えて環境に適応させる．これらの適応機能のコンポーネントを分散配置させ組み込みことによって，ネットワーク上に分散したシステムに適用可能であり，有効であると考えられる．しかし，大規模・複雑化している昨今のシステムにおいて，すべてのソフトウェアを考慮し恒久的に正常に稼働するシステムの設計方法やドメイン特性への適用は難しく，実際のシステムに組み込むにはまだまだ多くの課題がある．

### 2.2 MAPE ループの構成パターンとその問題点

さまざまな複雑かつ分散したシステムの特性に対応するために，MAPE ループを組み込んだ自己適応システムの構成パターンがいくつか提案されている．[6] では，MAPE の構成パターンとして Coordinated Control, Information Sharing, Master/Slave, Regional Planning, Hierarchical Control の 5 つのパターンが挙げられている． 以下にそれぞれのパターンの特徴を述べる．

Coordinated Control パターンは分散自己適応システムのための基本的な構成パターンである（図 2）．分散した各システムに MAPE コンポーネントを分散させ配置し，同じタイプのコンポーネント同士が通信，相互作用して自己適応を行う（M 同士, A 同士, P 同士，E 同士で相互作用する）．MAPE ループを分散させることにより，各コンポーネント同士で局所管理ができ，ロバスト性の向上，単一障害の回避ができる．しかし，各分散したコンポーネント同士の関係を詳細に把握していなければならず，スケーラビリティが低下してしまうおそれがあり，また，通信コストが高く，オーバーヘッドを起こしてしまう可能性がある．

Information Sharing パターンは，M（監視）コンポーネント同士のみが通信し，他のコンポーネントの通信を制限して適応を行う構成パターンである（図 3）．M(監視) 以外のコンポーネントは調整する必要がないため，オーバーヘッドを起こしにくく，システム全体を把握する必要がないため，巨大なソフトウェアシステムにも適しており，Coordinated Control パターンよりスケーラビリティも高い [7]．しかし，M（監視）による環境情報の共有だけによる適応・調整するため，適応精度・範囲に制限がかかる可能性がある．

Master/Slave パターンは M（監視）および E（実行）コンポーネントから構成される多数の Slave コンポーネントと，A（分析）および P（計画）コンポーネントから構成される Master コンポーネントから適応ロジックが構成されるパターンである（図 4）．分散した M（監視）コンポーネントは分析 (A) コンポーネントと，P（計画）は分散した E（実行）コンポーネントと通信し合い適応を行う．Master/Slave パターンは，分散したシステムから監視データを集め効率よく分析し，分散したシステムに同じ適応指示を出すため，複数のマシーンが並列して同じ動作を行うのシステムの自己適応に適している．A（分析) および計画 (P) コンポーネントを中央に配置し設計することは，効率的であり，分析・計画対象が特定しやすく，実装が容易にできる．しかし，Master コンポーネントは複数の slave コンポーネントを管理するため，アクセスが集中した場合負荷がかかってしまい，ボトルネックになる恐れがある．

Regional Planning パターンは，各分散したシステムに M（監視），A（分析），E（実行）コンポーネントを配置し，集中管理を行うシステムに P(計画) コンポーネントを配置

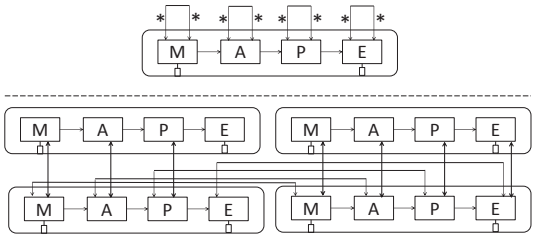



**図 2** 上:Coordinated Control パターン．下: パターン例

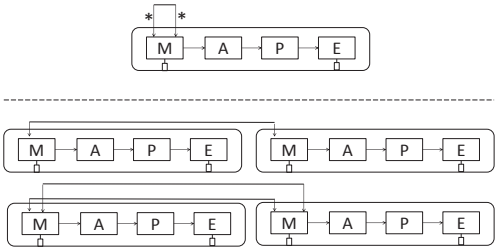



**図 3** 上:Information Sharing パターン．下: パターン例

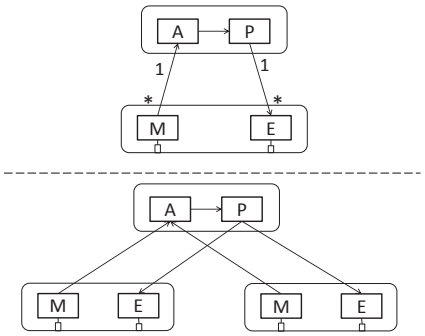



**図 4** 上:Master/Slave パターン．下: パターン例

し計画を行い，自己適応する構成パターンである（図 5）．データの局所管理を行うことで，各コンポーネント同士が通信し合うデータ量を減らすことができる．しかし，情報を集中させ分析するため，相当な P(計画）コンポーネントを配置したマシーンに負担がかかり，オーバーヘッドが発生してしまう恐れがある．このパターンのフレームワークとして，サービス指向システムにおける MOSES フレームワーク [8] が提供されている．

Hierarchical Control パターンは，階層ごとに MAPE コンポーネントを配置した構成パターンである（図 6）．階層構造はより高いレベルの MAPE ループが可能であり，より詳細な適応を対象とすることが可能である．しかし，階層ごとに制御することは難しく，達成すべき複数のゴールが互いに干渉し合う場合，MAPE コンポーネントの配置，関係性の割り当てを決定することは難しいとされている．そのため潜在的なトレードオフが多く発生する可能性が高く，設計・管理が非常に難しくなるおそれがある [8]．

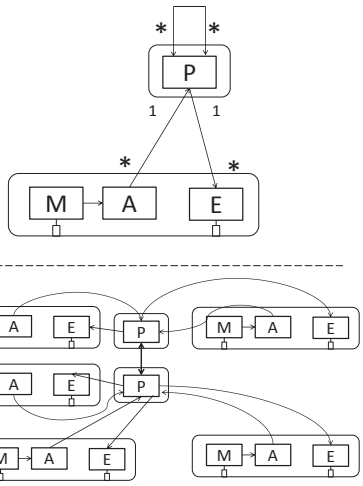



**図 5** 上:Regional Planner パターン．下: パターン例

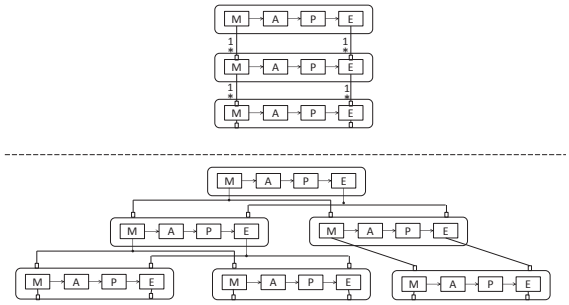


**図 6** 上:Hierarchical Control パターン．下: パターン例

## 3. オンラインゲームにおける課題

オンラインゲームとは，コンピュータネットワークを介して専用のサーバや他のユーザのクライアントマシン（PC，ゲーム機など）と接続し，同じゲーム進行を共有することができる，ソフトウェアを含むサービスを指す [9]．オンラインゲームは世界中に人気がある娯楽コンテンツであり，商用にも広く活用されている．長くユーザに活用してもらうためには，飽きさせないコンテンツの提供，多人数による同時接続が可能，タイムラグの軽減，セキュリティの高さ，恒久性，ユーザビリティの向上が求められる．しかしこれらの要素は，コストや時間という点からトレードオフ関係にあり，まだまだ課題が多く存在する．特にタイムラグの軽減の実現は，リアルタイム性のあるオンラインゲームにおいて，コンテンツのクオリティに関わる重要な要素である．オンラインゲームのタイムラグは，予測しえない情報の処理，各クライアントの利用マシンの性能，地理的距離などの差により避けられない問題である．

本研究では，C/S 型（クライアント・サーバ）におけるリアルタイム性のあるオンラインゲームにおいて，タイムラグの軽減を対象とした，分散自己適応システムの MAPE ループ構成パターンの提案を行う．本提案 MAPE ループ

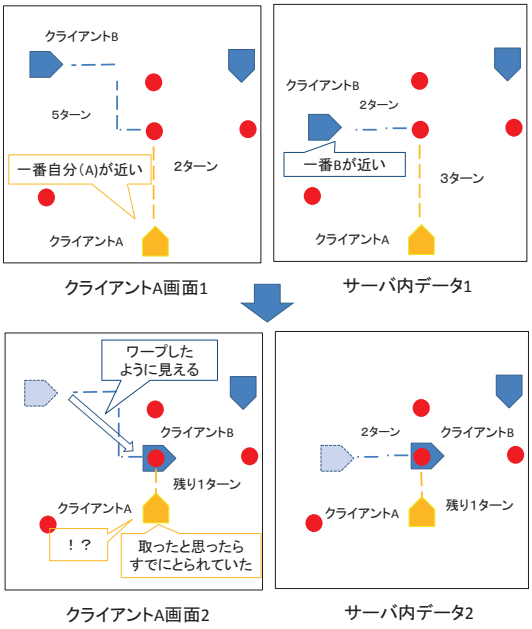



図 7　オンラインゲームにおける同期ずれによる問題例

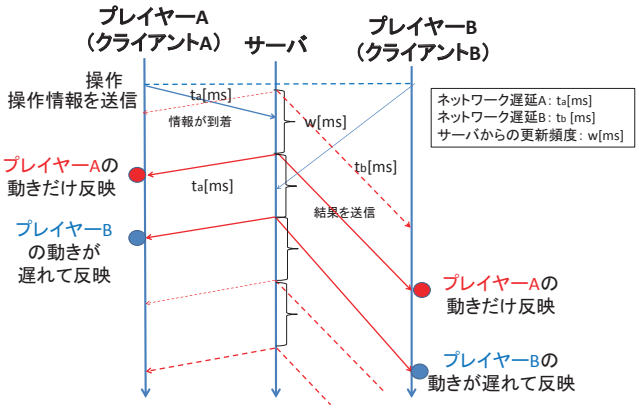


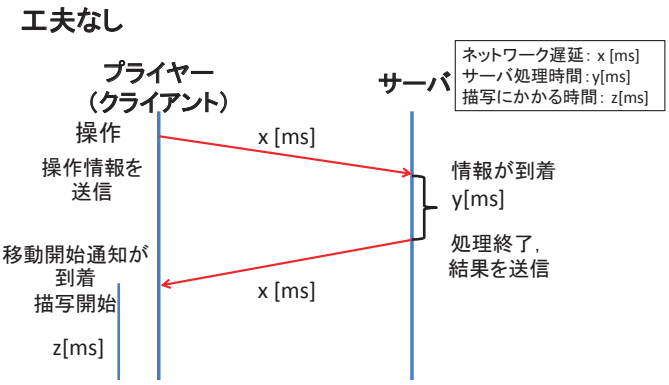

図 8　クライアントによる反映の差

構成パターンを適用することにより，分散した各クライアントの通信環境の違いを考慮した自己適応の実現を目指す．

### 3.1　タイムラグによる問題

オンラインゲームでいうタイムラグとは，ネットワークによる通信遅延のことを指す．このタイムラグが頻繁に起こってしまう環境では，プレイヤー同士のゲーム環境の差が大きく影響し，不公平なゲームになってしまう．タイムラグの原因として，ネットワークにおける距離が遠いためにタイムラグが発生してしまうケースが多い．タイムラグはオンラインゲームの快適さを示す重要な指標であり，通信速度は光の速さを超えられないため，常に課題になっている．このタイムラグが発生していると，画面がカクカクした動きをしたり，同期のずれが原因で突然オブジェクトがワープしたりという現象が起こる [10]．

図 7 はオンラインゲームにおける同期のずれによる問題



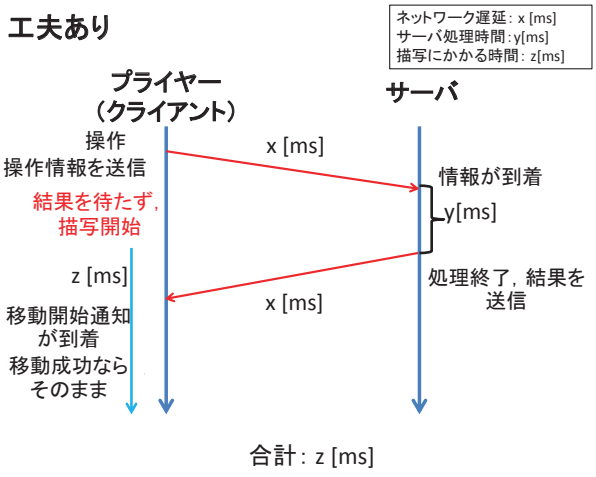




図 10　画面表示ラグを改善する工夫あり

例を表している．図 7 は赤い点をクライアント同士が奪い合ってスコアを競うオンラインゲームである．ここでクライアント A は，サーバとの距離が遠くタイムラグは大きく，各クライアントのステータス情報の更新が遅いとする．このような状況下では，クライアント A が赤い点を取りにいくために最適な最短距離を選択したとしても（図 7 上），到達時には他のクライアントにその赤い点を取られていしまうという現象が発生する可能性がある（図 7 下）．

C/S 型（クライアント・サーバ）のオンラインゲームでは，定期的にサーバからクライアントに定期的にステータス情報を送っている [10](図 8)．そのため，各クライアントのタイムラグに差があると，クライアントにサーバデータから送られてくるステータス情報を受け取るタイミングにズレが生じる．このことにより，クライアントがそれぞれ異なるステータスデータを持つ時間帯が発生することによる不平等な状況下を生み出し，ゲームバランスが保てなくなる可能性がある．

### 3.2 オンラインゲームのタイムラグに対する対策の現状

オンラインゲームにおけるタイムラグの対策の一つとしてとして，画面表示を方法を変えることによる改善方法が行われている [9]．ネットワークによる遅延を x[ms] とし，実際の描写処理に y[ms]，サーバ側での処理に z[ms] が必要な場合，ブラウザ式ゲームでは図 9 のような順番でクライアント側操作が反映する事象が起こる．このような処理方法を行うと，処理が反映されるまで (2x + y + z)[ms] かかる．図 10 では，サーバからの応答を待たずに，クライアント側で描写処理を行っている．そのためサーバが正常に操作が成功した場合，z[ms] で全体処理が完了する．これにより，クライアント側は 9 より 10 のほうがスムーズに操作ができていると感じることが出来る．しかしこのような方法の問題点として，ゲーム状態に不整合が生じてしまう問題が発生させないために，サーバ側から操作の失敗が返ってきた場合，瞬間的に元いた位置に戻す必要がある．また，他のクライアントの操作に依存するゲームである場合，クライアント間で状態が共有されないため，いつも有効であるとは言えない．他にもオンラインゲームのタイムラグに対する対策として，予めネットワークのタイムラグが大きい通信状況を想定し，クライアントとサーバ間でのデータのやり取り・更新の周期を調整し，ネットワークのタイムラグによるゲームの影響を抑える設計方法がとられている．しかし，この方法はあらかじめ想定された条件下による調整方法であり，動的に各クライアントの状況に対応した，クライアント・サーバ間で通信周期の調整は行われていない．

また，ゲームの快適性を損なう通信障害の原因として通信遅延によるタイムラグの他に，通信帯域やパケットロスによって発生する通信障害が挙げられる．パケットロスは，サーバが混雑し負担がかかるときに発生しやすく，クライアント側では解決できない．そのため対象とするゲームマシンによってゲームの帯域設計をする必要があり，パケットロスをリカバリするために定期的にまとめてパケットを送信，サーバ負荷を軽減するために通信量を節約する工夫がなされている [10]．

## 4. 提案パターン

2 章では MAPE ループ構成パターンにおける，各構成パターンの特徴，3 章ではオンラインゲームにおける問題点，タイムラグ問題の重要性を示した．

本章では 2 章で示した構成パターンを参考にし，分散したクライアントの異なる環境の違いを考慮した，高度な相互作用性が求められる C/S 型オンラインゲームの自己適応化に向けた MAPE ループ構成パターンを提案する．

### 4.1 提案パターンの概要

図 11 は，本研究の提案 MAPE ループ構成パターンを示したものである．図 11 上部の図は，提案パターン，下部はパターンの具体的な例を示す．本パターンは，オンラインゲームのようなクライアント・サーバモデルのシステムにおいて適用できる．クライアントノードに M（監視），A（分析）コンポーネント，オンラインゲームシステムを管理するサーバノードに P（計画），E（実行）を配置することによって，クライアント上でなくては得られないデータ・データの分析を元に，サーバが各クライアントの環境を考慮したゲームの進行を管理することができる．

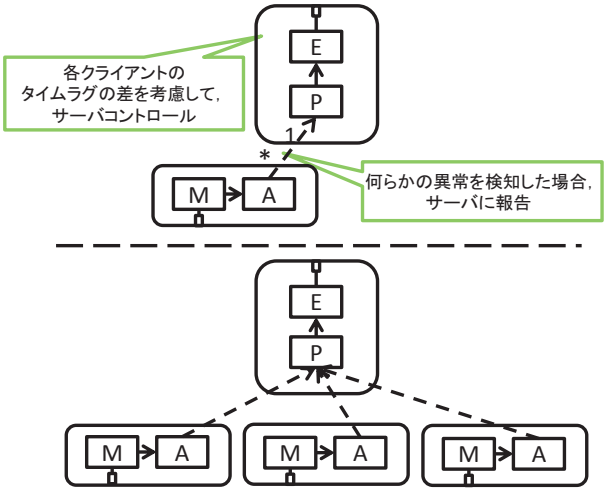



**図 11** 上:提案 MAPE ループ構成パターン．下:提案パターン例

### 4.2 有効性

本セクションでは，提案するオンラインゲームに適した MAPE ループ構成パターンの有効性について述べる．．本パターンの特徴は，C/S 型システムにおいて，クライアント側の監視データを元に分析し，サーバ側で計画を立て，構成を切り替える点である．こうすることにより，既存の MAPE ループ構成パターンではできなかった分散したクライアントがサーバに依存するシステムにおいて，クライアント側の問題をサーバ反映させることのできる自己適応が可能となる．

### 4.3 パターン適用例

本研究の提案 MAPE ループ構成パターンは，ネットワーク上におけるオンラインゲームのような高度な相互作用性が求められる C/S 型システムにおける自己適応に適している．図 12 は C/S 型オンラインゲームシステムに本提案パターンを適用した例である．各クライアント，ゲームサーバそれぞれに，サブシステムとしてノード（M，A），ノード（P、E）が配置，組み込まれている．オンラインゲームシステムは，すべてのクライアントが同じタイミングで動作が反映され，ステータス情報を持ち動作することにより，プレイヤー同士のゲームバランスが保たれる．本自己

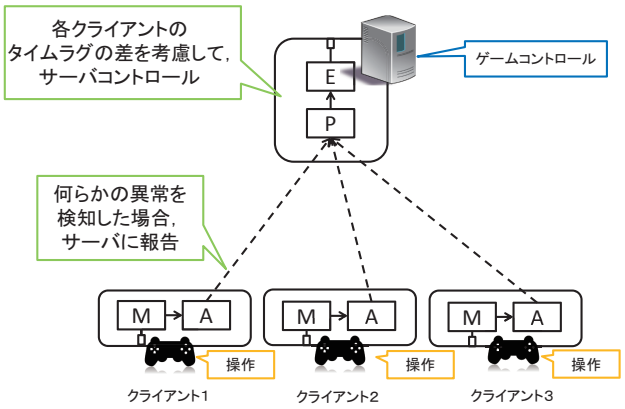



図 12　提案パターンを組み込んだ C/S 型オンラインゲームシステム

表 1　実験 1，2 におけるクライアントが通信にかかる時間

| | client1[ms] | client2[ms] | client3[ms] |
|---|---|---|---|
| 実験 1 | 8400(+800) | 7600(+0) | 7600(+0) |
| 実験 2 | 8400(+800) | 7600(+0) | 9600(+1200) |

表 2　実験における自己適応における更新周期の設定

| | 更新周期 [ms] |
|---|---|
| a | 2550 |
| b | 3050 |
| c | 2000 → 3200↘ |
| d | 2000 → 3800↘ |

適応シナリオは，定期的にサーバから各クライアントに定期的にステータス情報を送る更新周期を，各クライアントの通信環境を考慮し，適した更新周期に動的に変化させる (図 13)．これにより常に同じステータス情報を各クライアントが取得することが可能となる．

図 14 は本シナリオにおける MAPE ループの流れを示す．MAPE コンポーネントは，M（監視）および A（分析）の活動によって蓄積された情報に基づき，P（計画）は計画を行い，E（実行）が計画に従い実行をすることにより適応する．M（監視）は，図 7 のような矛盾がクライアントごとに発生していないか監視する．A（分析）は M（監視）によって集められた情報を元に，矛盾が発生している原因，各クライアントのタイムラグの差を分析する．P（計画）は A（分析）の分析結果をもとに，常に同じステータス情報をクライアントが取得できるように，サーバから各クライアントに定期的にステータス情報を送る周期を計画する．E（実行）は P（計画）の立てた計画を元に，定期的にステータス情報を送る更新周期を変更する．

## 5.　評価実験

本章では，提案 MAPE ループ構成パターンを適用したシステムについて説明し，本提案 MAPE ループ構成パターンを組み込んだ自己適応システム（a，b) と，組み混んで

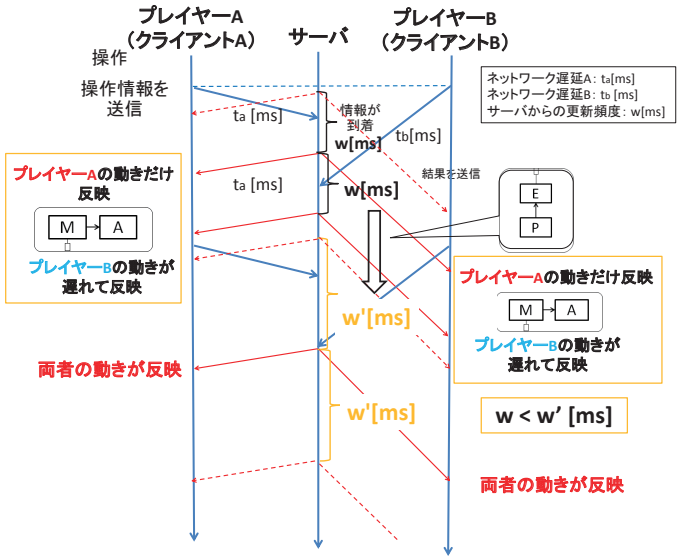



図 13　更新周期変更による自己適応

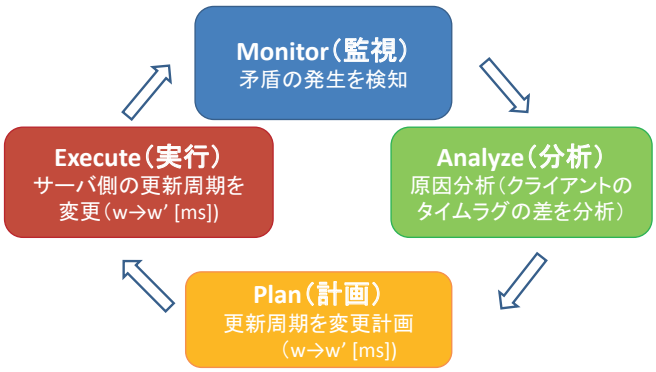



図 14　パターン適用例における MAPE ループ

いないシステム (c，d) システムの各クライアントのタイムラグの差による異なるデータ共有（矛盾）の回数を比較することによって有用性を評価する．

### 5.1　提案構成パターン適用システム

C/S 型簡易オンラインゲームシステムを構築し，本提案 MAPE ループ構成パターンを組み込み実装した．各クライアントは同じ動作のデータをサーバに送り，サーバはそのデータを処理し，ステータス情報を一定の周期で各クライアントが取得できるようにする．各クライアントが持つ自他のステータスに差が発生をシステム組み込まれた M（監視）が認識し，その情報を元に A（分析）がクライアントのタイムラグの差を分析する．サーバに組み込まれた P（計画）が更新周期の決定，E（実行）が実行することによって，自己適応を行う (図 13)．

### 5.2　実験内容

本研究では，クライアント同士が異なるデータ共有（矛盾）がより多く発生するように，各クライアントにタイム

ラグの差をプログラミング上で sleep time 関数により持たせ，実験を行った．クライアントが通信にかかる時間と更新周期はそれぞれ表 1，表 2 に示す．更新周期 a，b はそれぞれ更新周期 c，d の平均である．異なるクライアントが通信にかかる時間を持つ client1，client2，client3，異なる更新周期 a〜d において計 8 回実験を行った．

### 5.3 結果・考察

実験結果を表 3 は示す．図 15 は各実験 a，b，c，d において，本提案 MAPE ループ構成パターンの実装により変更された，サーバからの更新周期の変動グラフである．

更新周期を変更させることによるプレイヤー間の公平性とゲームの快適性はトレードオフ関係にある．更新周期を大きくするほどプレイヤー間のタイムラグの差による矛盾は少なくなるが，ゲームの進行の快適性は失われる．そこで自己適応を行う更新周期 c，d とそれぞれの平均更新周期 a，b を比較することにより，トレードオフ関係を考慮した，本提案 MAPE ループパターンの有用性に関する実験を行った．なお，本研究では厳密なゲームの快適性は考慮せず，各クライアントにおいて異なるデータ共有（矛盾）の回数に焦点を当てている．表 3 より実験 1 おいて，更新周期 a，b，c，d はそれぞれ矛盾検出数は 60，45，48，39 個であり，$a > c$，$b > d$ と矛盾検出数が減っているのがわかる．また，実験 2 においても更新周期 a，b，c，d はそれぞれ矛盾検出数は 54，45，45，36 個であり，$a > c$，$b > d$ と矛盾検出数が減っているのがわかる．このことから同じ平均更新周期において，本提案 MAPE ループ構成パターンを組み込むことにより，矛盾検出数を減らすことができ，有用性があることを確認できた．

しかし，本実験は実験環境におけるタイムラグや測定誤差，オンラインゲーム環境における最適な更新周期の設定を考慮していない．また矛盾を検知してから，一定のサイクルによる更新周期の変更を行っている．更新周期の変更を学習させることによって，動的に更新周期の変更操作ができ，より優れた自己適応が可能であると考えられる．

また M（監視）における監視対象の設定をクライアントが持つ位置に関するステータスデータに限定している．オンラインゲームにおいて，タイムラグの差による矛盾は他にもさまざま想定される．クライアント・サーバ間の上り下りの通信速度の考慮，サーバ負荷，クライアントマシーンスペックの差，ゲームコンテンツ独自のタイムラグによる影響，自己適応メカニズムによって発生するトレードオフ問題など様々な要素が想定させる．それらを考慮した自己適応を行うことによって，より適応範囲が広くなると考えられる．

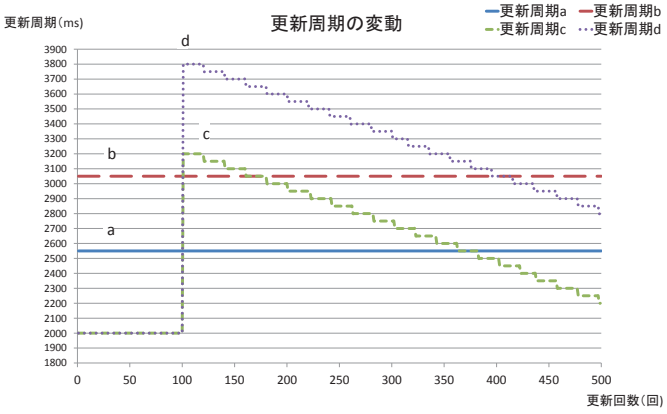



**図 15** 実験における更新周期の変動

**表 3** 自己適応による矛盾検出数の変化

| 実験 1 | 更新周期 a | 更新周期 b | 更新周期 c | 更新周期 d |
|---|---|---|---|---|
| 矛盾検出数 (個) | 60 | 45 | 48 | 39 |

| 実験 2 | 更新周期 a | 更新周期 b | 更新周期 c | 更新周期 d |
|---|---|---|---|---|
| 矛盾検出数 (個) | 54 | 45 | 45 | 36 |

## 6. まとめと今後の課題

本研究では，オンラインゲームにおける課題と現状について着目し，C/S 型オンラインゲームに適した新しい MAPE ループ構成パターンの提案を行った．C/S 型簡易オンラインゲームに，本提案 MAPE ループ構成パターンを組み込み実装し，MAPE ループを組み込んでいないシステムとの比較実験をし，有用性を評価した．

しかし，本実験は実験環境におけるタイムラグや測定誤差，オンラインゲームにおける最適な更新周期の設定を考慮していない．矛盾を検知してから，一定のサイクルによる更新周期の変更を行っている．また，想定している通信時間，タイムラグの設定時間をより矛盾が発生しやすくするため，長く設定している．これらを改善するため，今後はより実環境での適用ができるよう通信速度・タイムラグの設定時間を短くし，更新周期変化と矛盾の回数とのより詳細な関係を明らかしていきたい．さらに更新周期の変更を学習させることによって，動的に更新時間の変更を行うことのできる，より優れた自己適応メカニズムが構築していきたい．

今後の課題として，オンラインゲームクライアント・サーバ間の上り下りの通信速度の考慮，潜在的なトレードオフ，不特定多数のクライアントにおける適用，実用性のある M（監視）における監視対象の設定・適応，サーバ負荷，各クライアントのマシンスペックなども考慮し，取り組んでいきたい．また，既存 MAPE ループ構成パターンとの比較，ほかの種類のゲームにおいて組み込み適用させることにより，本提案 MAPE ループ構成パターンの有用性を評価す

る予定である．C/S 型オンラインゲームの他にも，あらゆるシステムに組み込み有用性を評価することにより，適用できるシステムドメイン範囲を確認していきたい．

謝辞

本研究は JSPS 科研費 24300005，23500039，25730038 の助成を受けたものです．

## 参考文献

[1] Rogerio de Lemos, Holger Giese, Hausi A. Muller, Mary Shaw, et al.,.: *Software Enginnering for Self-Adaptive Systems: A Second Resarch Roadmap*,Dagstuhl Seminar Proceedings 2011, pp. 1-16 , 2011.

[2] Betty H.C Cheng, Rogerio de Lemos, Holger Giese, Paola Inveradi, Jeff Magee, et al,: *Software Engineering for Self-Adaptive Systems: A Reaearch Road Map*,Dagstuhl Seminar 2008, pp. 1-26 , pp. 1-26 , 2008.

[3] Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), pp. 41-50, 2003.

[4] Dobson, S., Denazis, S., Fernndez, A., Gati, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.*A survey of autonomic communications*, ACM Transactions Autonomous Adaptive Systems (TAAS) 1(2), 223?259 , 2006

[5] IBM RedBooks: On Demand Operating Environment: Managing the Infrastructure (SG24-6634-01, June 2005, Second Edition), 2005

[6] Danny Wenyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Reffaela Mirandola, Cristian Prehofer, et al, *On Patterns for Decentralized Control in Self-Adaptive Systems*, Software Engineering for Self-Adaptive Systems II Lecture Notes in Computer Science Volume7475, pp. 76-107, 2013.

[7] Weyns, D., Iftakhir, M.U., Malek, S., Andersson, J.: *Claims and supporting evidence for self-adaptive systems: A literature review.* In:Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2012. ACM, New York, 2012.

[8] Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F.: *Adaptive Management of Composite Services under Percentile-Based Service Level Agreements.* In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp.381-395. Springer, Heidelberg , 2010

[9] 中嶋 謙吾，オンラインゲームを支える技術〜壮大なプレイ空間の舞台裏〜，技術評論社，（2011）

[10] 節政 暁生, ネットワークゲームの仕組みとゲームデザイン，Computer Entertainment Developers Conference2010，（2010）