自律的ネットワーク運用のための管理知識の利用手法に関する研究

平成29年度 博士学位論文

自律的ネットワーク運用のための
管理知識の利用手法に関する研究

東北大学大学院 情報科学研究科
情報基礎科学専攻 博士課程後期3年の課程
コミュニケーション論講座　木下・北形研究室

B5ID1004 谷村 優介

目 次

図 目 次

表 目 次

第1章　序論

第 1 章では，本研究を行うに至った背景について概説し，本研究の対象である，知識型ネットワーク管理システムを用いた自律的なネットワーク運用について述べるとともに，既存研究に対する本研究の位置づけを示す.

1.1　背景

1.1.1　ICT サービスの普及とネットワークシステムの管理

ICT（Information and Communication Technology）技術は，コンピュータネットワーク（以下ネットワーク）を介してサービスを提供する技術であり，多くの企業や組織が，ICT 技術によって産み出される ICT サービスを基盤とした活動を行っている．そして，企業内の活動のみならず，従来からのインターネット利用者数・人口普及率の増加に加え，高い性能と多様な機能をもつスマートフォン・タブレット端末の急速な普及により，場所を問わず高品質な ICT サービスを利用することが可能となるなど，今や ICT 技術は人々の日常生活においても欠かすことのできない存在となった [1].

また，仮想化技術とネットワーク技術の高度化により登場したクラウド・コンピューティング技術は，既に普及期に入っており，IaaS（Infrastructure as a Service）や PaaS（Platform as a Service），SaaS（Software as a Service）のような利用形態に基づき，ICT サービスを提供するための環境を容易に構築することが可能となった．計算資源およびネットワーク機能の仮想化により，ハードウェア構成に束縛されずに配置や設定の変更が可能となり，サービス利用者の需要の量的・質的

や変化に対して迅速に対応可能な，より柔軟性の高い ICT サービスが実現できる．クラウド・コンピューティング技術の進展によって，既存の ICT サービスの利便性の向上だけでなく，あらゆる分野の ICT 化および ICT サービスの利活用が，今まで以上に促進されている [2].

さらに，以前より推進されてきた，ユビキタス社会の実現に向けた要素技術の進展を背景として，IoT（Internet of Things）や IoE（Internet of Everything）といった新たなパラダイムが登場し，近年注目を集めている [3]．近い将来にその到来が期待されている IoT/IoE 時代においては，スマートフォン・タブレット端末だけでなく，情報家電の普及等により，身の回りのあらゆるモノが計算機能を持ち，ネットワークに接続されると予想される．身の回りのモノがネットワークに接続され，相互に連動することで，身の回りから発生する多種多様な情報に新たな価値が見いだされるとともに，これまでにない新たな ICT サービスが創出されると考えられる．また，これまで一方的に ICT サービスを受け取るだけであったサービス利用者も，IoT/IoE 技術の普及した世界では，利用者自身が発する情報や自身の周辺で発生し続ける情報に価値を見いだし，自分自身で利用するため，あるいは，他者に提供するために ICT サービスを創りだし，そのサービスの提供者となることが期待される．

上述の背景より，今後のクラウド・コンピューティング技術のさらなる成熟，そして，IoT/IoE 技術の進展と普及によって，多種多様な ICT サービスが創出され，加速度的に ICT サービスの提供者が増加すると考えられる．ICT サービスの提供者は，ICT サービスを提供するためのハードウェア・ソフトウェアから構成される，ネットワークシステムの管理を行わなければならない．しかしながら，ネットワークシステムの管理を行うネットワーク管理者にかかる負担の増大や，それにともなう人材不足は以前より大きな問題となっており，また，IoT/IoE 技術の進展と普及にかかる課題としても同様に，計算機器やネットワーク機器の管理を円

滑に行うことのできる人材の不足が指摘されている [2, 4]．ネットワークシステムの円滑な管理には，管理対象である計算機器やネットワーク機器に関する深く広い知識が求められることに加え，豊富な管理経験に基づく管理スキルが必要とされる．よって，知識や管理経験の少ない初級管理者にとって，ネットワークシステムを円滑に管理し，ICT サービスを安定的かつ継続的に維持することは非常に難易度の高いタスクである．さらに，前述のとおり，近年の ICT サービスの構築に関する技術の進展は目まぐるしく，その潮流の変化に対応するためには，熟練した管理者であっても常に新たな知識と経験を獲得し続けなければならない．

以上より，今後の ICT サービスのさらなる普及と進展，および，IoT/IoE 時代の到来に備えるため，ネットワークシステムを円滑に管理可能とするための，管理者支援の仕組みの開発が急務であると考えられる．管理者支援の仕組みには，初級管理者をネットワーク管理に参加可能とするため，そして，エキスパートな管理者の管理負担を軽減するため，管理知識と管理経験を補うことにより，管理者の知識負担と作業負担を軽減する機能が求められる．

1.1.2　ネットワークシステムの管理支援

ネットワークシステムの管理を支援するため，これまでネットワーク管理システム（Network Management System: NMS）に関して，様々な研究開発が行われてきた．近年のネットワークシステムの管理で一般的に使用されている NMS は，SNMP（Simple Network Management Protocol）やネットワークベンダー独自の管理プロトコル等に基づく，監視機能を中心とした設計となっている（図 1.1） [5, 6, 7]．監視機能を中心とした NMS は，ネットワークシステムを構成するサーバやネットワーク機器からの情報取得や，バッチジョブの実行など，イテレーションをともなう管理作業の自動化による管理者負担の軽減を主な目的としている．このような NMS には，管理プロトコルによりテキストベースで得られるネットワークシス

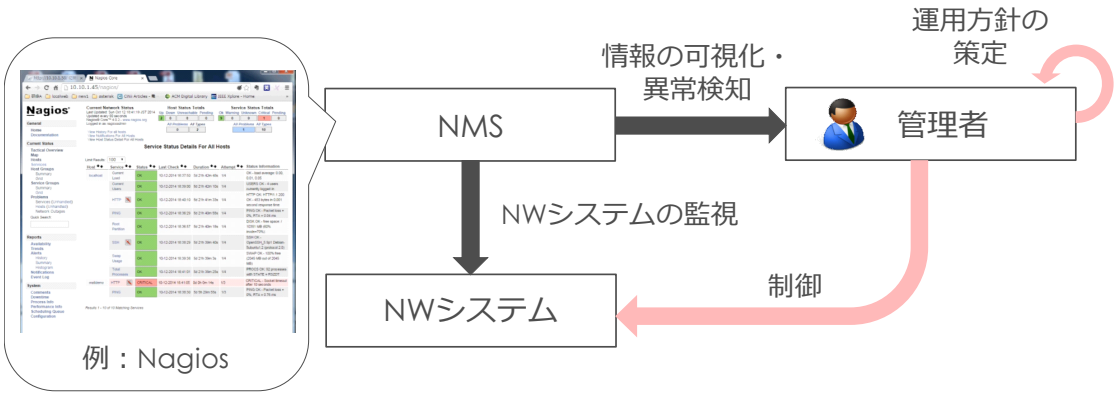


図 1.1: 監視機能を中心とした NMS による管理者支援

テムの情報を処理・加工し，グラフィカルに表示する一連の作業を自動化するための可視化システムや，事前に設定した管理対象の正常状態に関する判定基準と比較する一連の作業を自動化するための異常検知システムなどが含まれる．

しかしながら，ICT サービスを安定的かつ継続的に提供するためには，ネットワーク管理者は NMS から得られる情報に基づいてネットワークシステムの状態を把握するだけでなく，さらに，状態に応じた運用方針を導き出し，ネットワークシステムに適用する必要がある．ネットワークシステムの状態の見極めや，状態に基づく運用方針の策定と適用は，ネットワーク管理者に知識と経験が最も必要とされる場面である．例えば，可視化システムによりグラフィカルに提示された管理対象システムの情報の解釈，そして，異常検知システムにより検出された異常の発生場所や原因の特定には，管理者にさらなる知識と経験が要求される．よって，監視機能を中心とした NMS は，豊富な知識とスキル，そして管理経験を有する熟練した管理者とっては有用であるが，初級管理者がネットワークシステムの管理に参加するためには，より知的な支援が与えられることが望ましい．さらに，熟練した管理者にとっても，従来の管理者負担の増大に関する課題の解決に向け，管理作業の自動化の範囲を拡げるための，より高度な支援の仕組みが必要である．

1.1.3　知識型ネットワーク管理システムに基づく自律的なネットワーク運用

このような観点から，人間の熟練した管理者がもつ経験的・専門的な管理知識を NMS に付与することで，NMS の管理者支援能力を強化し，より知的な管理支援機能の実現，さらには，ネットワーク管理の自律化を目指す試みが行われている [8, 9]．本研究では，人間の管理者から与えられた管理知識を用いて自律的に動作する NMS を総称して「知識型ネットワーク管理システム（Knowledge-based Network Management System: KNMS）」と呼ぶ．

ネットワークシステムに自律性を付与するための管理知識には，管理者由来の知識やネットワークシステム由来の知識がある．管理者由来の知識は，ネットワークシステムの管理を行う際のエキスパートな管理者の振るまい，すなわち，管理業務に関する具体的な操作や手続き，考え方に関する記述を含む．そして，ネットワークシステム由来の知識は，管理対象のネットワークシステムの振るまい，すなわち，ネットワークシステムがもつ特性に関する記述を含んでいる．KNMS はこれらの管理知識に基づき，エキスパートな管理者と同様の方法で，かつ，管理対象のネットワークシステムの特性に合わせた，具体的な制御行動を実行することができる．ゆえに，自律的なネットワーク運用の実現を目指す KNMS において，管理知識は KNMS の自律性を司る最も重要な構成要素であると言える．

図 1.2 に KNMS の概要を示す．KNMS は管理対象とするネットワークシステムから情報を取得する機能だけでなく，ネットワークシステムの状態を見極め，運用方針を策定するための管理知識，そして，運用方針に基づいてネットワークシステムの制御を行う機能から構成される．KNMS は管理対象とするネットワークシステムから情報を取得するだけでなく，管理知識に基づいて，具体的にどのような管理操作を施すべきかを導出する点において，上述の監視機能を中心とした NMS と大きく異なる．KNMS の提供する，より知的な管理支援によって，知識と

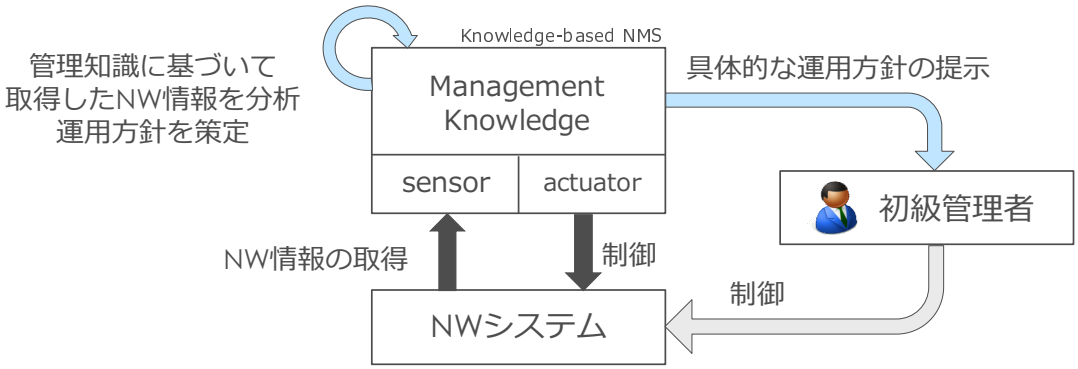


図 1.2: 知識型ネットワーク管理システムの概要

経験を補うことで，初級管理者のネットワークシステム管理への参加が可能となることが期待される．また，エキスパートな管理者にとっても，さらなる管理負担の軽減が期待される．

しかしながら，現在，KNMS は広く実用化・普及するまでには至っておらず，限定的な領域への適用にとどまっている．KNMS が有効にネットワーク管理者を支援するためには，管理対象のネットワークシステムのハードウェア・ソフトウェア的な構成に適合した管理知識が必要不可欠である．そのため，管理対象のネットワークシステムの構成が変化した場合には，知識を蓄える知識ベースの更新，すなわち，新たな管理知識の追加や，既存の管理知識の修正が必要となる．管理対象ネットワークシステムの構成に適合しない管理知識は，KNMS の管理者支援能力を大きく低下させるだけでなく，不適切な運用方針の導出と適用により，ICT サービスの安定的かつ継続的な提供を妨げる原因となる．知識に基づく自律システムの有効性は，その根幹たる管理知識の利用技術に大きく左右されることが指摘されており，KNMS もまた，管理知識の利用手法，すなわち管理知識の獲得・表現・適用にかかる技術が，その有効性を左右すると言える [10]．そのため，知識ベースの更新は容易な作業ではなく，管理者に大きな作業負担・心的負担を強いるものである（図 1.3）．

ゆえに，従来の KNMS は知識ベースを頻繁に更新する必要のないネットワーク

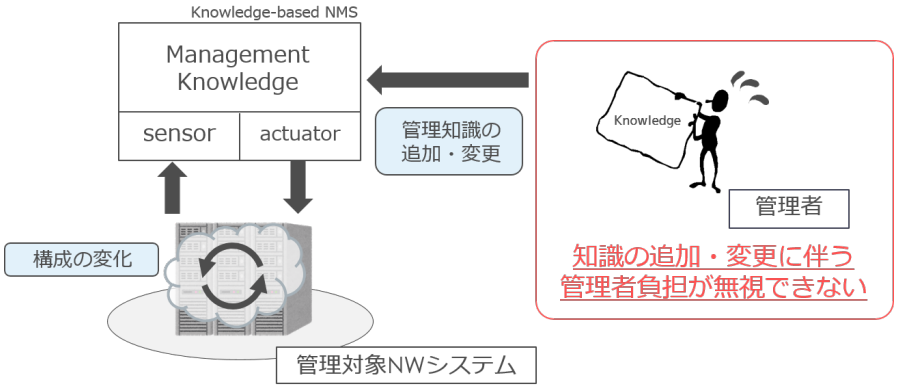


図 1.3: KNMS の変動するネットワークシステムへの適用に関する課題

システム，すなわち，構成の変化しないネットワークシステム，あるいは，どのように構成が変化するかをあらかじめ把握可能なネットワークシステムへの適用を前提として，限定的な領域におけるネットワーク運用の自律化にとどまっていた．

1.2　目的

1.2.1　本研究の目的

前節で述べたとおり，今後のクラウド・コンピューティング技術のさらなる普及と進展，および，IoT/IoE 時代の到来により，新たな ICT サービスの創出や ICT サービス提供者の増加が期待・予測される．ICT サービスを安定的かつ継続的に提供するためには，ICT サービスを提供するためのハードウェア・ソフトウェアから構成されるネットワークシステムを円滑に管理する必要がある．しかし，ネットワークシステムの円滑な管理には，管理者に豊富な管理知識と管理経験が要求されるため，以前より，ネットワーク管理にかかる人材の不足が問題とされてきた．そこで，今後の ICT サービスのさらなる普及と技術の進展に向けて，管理者支援の仕組みの開発が急務であることを述べた．

そして，ネットワークシステムの管理を支援するための仕組みである，NMS について述べるとともに，人間の熟練した管理者がもつ経験的・専門的な管理知識

を NMS に付与することで，より知的な管理支援機能の実現やネットワーク運用の自律化を目指す取り組みである，KNMS について述べた．しかしながら，従来の KNMS は，管理知識の追加や変更，すなわち，知識ベースを更新する際の管理者負担の大きさから，知識ベースを更新する必要のない，構成の変化しないネットワークシステムや，構成の変化を事前に把握可能なネットワークシステムへの適用を前提としていた．そこで本研究では，管理知識の利活用手法に関する課題を解決することで，変動するネットワークシステムに対しても広く適用可能な，より実用的な KNMS の実現を目的とする．

1.2.2　本研究の焦点

本研究では，変動するネットワークシステムに適用可能な KNMS の実現のため，KNMS の管理知識の利用手法に焦点を当てる．そして，変動するネットワークシステムを対象に含めた，自律的なネットワーク運用を実現するため，どのように KNMS を構築すべきかについて論ずる．その中で，熟練管理者の専門知識の蓄積と利用を支える知識ベースをどのように構築し，システム構成の変化にともない発生する，管理知識の追加・変更にどのように対応するかについて述べる．さらに，ネットワークシステムの構成変化が発生した際にも追加・変更する必要のない管理知識，すなわち，システム構成の変化に強い汎用的な管理知識をどのように獲得・記述すべきかについて述べる．

本研究の位置づけを図 1.4 に示す．管理者支援能力は NMS としてどれだけ管理者負担を軽減できるか，その貢献の度合いを表し，そして管理性は NMS そのものの維持・管理のしやすさを表している．前節にて述べた，監視機能を中心とする NMS は，ネットワークシステムの情報の可視化や異常検知に関する作業の自動化という用途が明確であり，機能がシンプルであるため，導入と設定が比較的容易であり，NMS 自体の管理にともなう管理者負担は小さく，その管理性は必ずしも

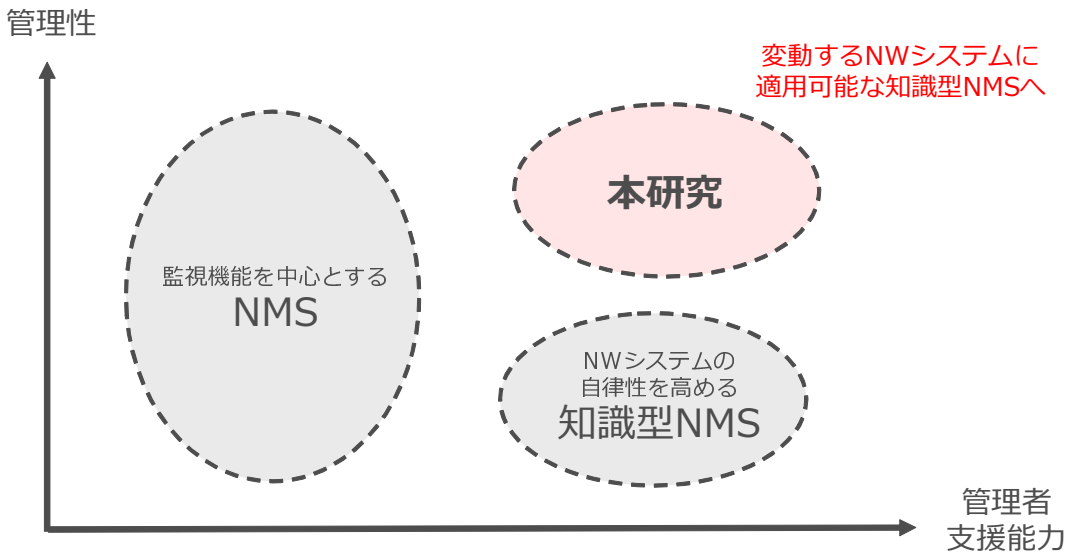


図 1.4: 本研究の位置づけ

低くはない．しかし，NMS から得た情報に基づいてネットワークシステムの状態を見極め，運用方針を策定するという知識と経験を必要とする作業は，管理者が行わなければならない．よって，監視機能を中心とする NMS による恩恵を受けるのは，既に豊富な知識と経験を有するエキスパートな管理者に限定される．一方で，ネットワークシステムの自律性を高め，管理者の知識とスキルを補う KNMS は，初級管理者のネットワーク管理への参加を促すなど，高い管理者支援能力をもつと言える．しかしながら，前節で述べたとおり，KNMS には自律システム特有の，管理知識の利活用に関する課題により，管理性は低く，変動するネットワークシステムへの適用を妨げる原因となっていた．本研究では，KNMS の自律性を司る管理知識の利活用に関する課題を解決することで，KNMS の管理性を高めることを狙いとする．そして本研究は，KNMS が変動するネットワークシステムにおいても知的な管理支援を提供することを可能とし，柔軟性と迅速性の高い ICT サービスの普及する今後の社会における，ネットワークシステムの管理者不足および，管理者負担の増大に関する課題の解決を図るものである．

1.3　本論文の構成

本論文の構成を図 1.5 に示す．本論文は全 5 章から構成される．

第 1 章「序論」では，本論文の研究背景について概説し，本研究の対象である，知識型ネットワーク管理システムを用いた自律的なネットワーク運用について述べ，既存の知識型ネットワーク管理システムに関する課題を述べた．そして，本研究の目的と本研究の焦点，既存研究に対する本研究の位置づけを示した．

第 2 章「知識型ネットワーク管理システムの現状と課題」では，知識型ネットワーク管理システムに関する従来の取り組み，特に，自律的な障害管理や性能管理などの運用管理タスクに関する取り組みに注目し，変動するネットワークシステムへの適用に際して発生する課題の詳細化を行い，さらに，本研究における課題解決のアプローチについて述べる．

第 3 章「知識型ネットワーク管理システムによる自律的な障害管理」では，知識型ネットワーク管理システムを用いた障害管理タスクの自律化に焦点を当て，熟練管理者の専門知識の蓄積と利用を支える知識ベースの構築法について検証し，知識のモジュール化を導入したサービス指向型の知識管理手法を提案する．

第 4 章「知識型ネットワーク管理システムによる自律的な性能管理」では，知識型ネットワーク管理システムを用いた性能管理タスクの自律化に焦点を当て，ネットワークシステムの全体的な運用状態を把握し，この結果を運用管理タスクに反映させる新しい手法を提案し，システム構成の変化に強い汎用的な管理知識をどのように獲得・記述すべきかについて述べる．

第 5 章「結論」では，本論文のまとめと本研究の貢献について述べ，今後に残された課題について述べる．

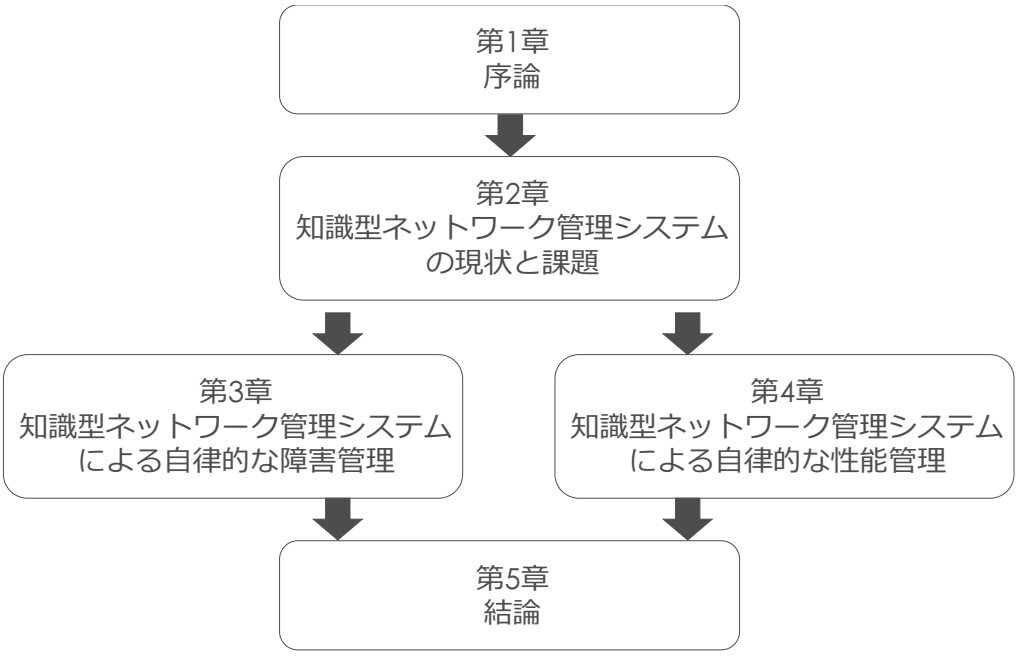

第1章
序論
第2章
知識型ネットワーク管理システム
の現状と課題
第3章
知識型ネットワーク管理システム
による自律的な障害管理
第4章
知識型ネットワーク管理システム
による自律的な性能管理
第5章
結論

図 1.5: 本論文の構成

第2章　知識型ネットワーク管理システムの現状と課題

第 2 章では，知識型ネットワーク管理システムに関する研究を取り上げ，特に，自律的な障害管理や性能管理などの運用管理タスクに関する取り組みに注目し，変動するネットワークシステムへの適用に際して発生する課題の詳細化を行い，さらに，本研究における課題解決のアプローチについて述べる．

2.1　知識型ネットワーク管理システム

2.1.1　自律的なネットワーク運用

自律的なネットワーク運用の実現に向けた，KNMS に関する代表的なアプローチとして，Autonomic Network Management がある [9]．Autonomic Network Management は生物の自律神経系のはたらきに着想を得た，Autonomic Computing [8, 11] の考え方をネットワークシステムの管理に応用した概念である．Autonomic Computing の考え方にしたがって作られたシステムは，MAPE-K（Monitor-Analyze-Plan-Execute over a Knowledge）ループと呼ばれる自律制御機構を持ち，自らの状態を正常に保つための自律性に関する特性を獲得する（図 2.1）．以下に，代表的な 4 つの特性（Self-CHOP properties）を示す．

Self-Configuring

自律的にシステムを構築し，さらにその構成を変化させることで，状況に合わせた構成をとることができる．

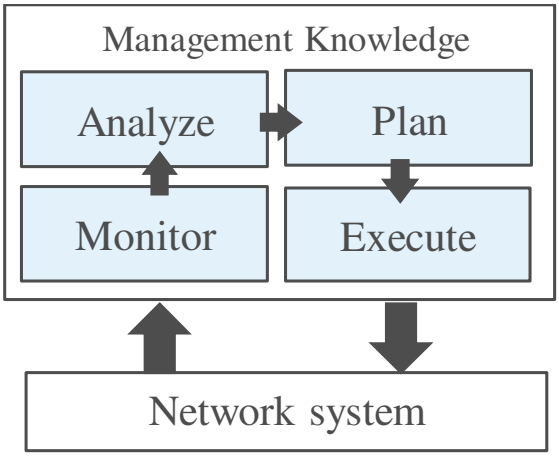


図 2.1: MAPE-K 自律制御ループ

Self-Healing

システムに障害が発生した場合に自律的に原因を特定し，対応策を施すことでシステムを修復することができる．

Self-Optimizing

自身の状態を的確に把握し，より効率的に動作するため，自律的にシステムの設定を変更することができる．

Self-Protecting

外部からの攻撃や侵入など，システムの円滑な運用を妨げるものから身を守ることができる．

以上で取り上げた自律性を実現するための 4 つの特性は，自律的なネットワークシステムの管理を目指す KNMS の取り組みにおいて，従来のネットワークシステムの管理モデル [12] と以下のように対応づけることができる．

自律的な構成管理

ICT サービスの提供に必要な計算機能とネットワーク機能を組み合わせ，自律的にネットワークシステムを構築することができる．

自律的な障害管理

発生した障害の原因を特定し，適切な対応策を導出・適用することで，自律的にネットワークシステムを修復し，継続的に ICT サービスを提供することができる．

自律的な性能管理

ネットワークシステムの運用状態を的確に捉えることで，自律的にネットワークシステムの調整を施し，安定的に ICT サービスを提供することができる．

自律的なセキュリティ管理

ICT サービスに対する外部からの脅威を検出することで，自律的に処置を施し，ネットワークシステムの円滑な運用を妨げるものから身を守ることができる．

KNMS が実現を目指す以上の特性は，互いに関係性があり，必ずしもそれぞれ独立した機能やシステムとして設計・実装される訳ではないが，本研究では特に，自律的な障害管理と自律的な性能管理，これら 2 つの特性に着目する．障害管理と性能管理は，ICT サービスの運用フェーズにおける管理者の主な日常業務であり，管理者の経験的な知識が最も活用される管理タスクである．さらに，ICT サービスの構築後その構成が変化することの少ない固定的なネットワークシステムにおいては，ライフサイクルのほぼ大半を運用フェーズが占める．ゆえに，KNMS を用いた従来の管理者支援に関する取り組みでは，運用フェーズにおける主な管理タスクである障害管理と性能管理が主な焦点であったと考える．よって次項より，自律的な障害管理，および，自律的な性能管理の実現に向けた KNMS の取り組みについて述べ，変動するネットワークシステムへの適用時に発生する課題を詳細化する．

2.1.2　自律的な障害管理に関する取り組み

障害管理の自律化に関して，以前よりエキスパートシステムをはじめとした様々な取り組みが行われてきた [9, 13, 14, 15, 16, 17, 18, 19, 20]．障害管理における管理者の主なタスクは，管理対象のネットワークシステムに障害が発生し，ICT サービスの提供に問題が発生した場合に，障害の発生箇所や原因を特定し，適切な対応策を考え出し，その対応策をネットワークシステムに施すことである．そのため，障害管理の自律化，すなわち，KNMS が以上の管理者のタスクを代行するためには，障害原因の特定や対策案の導出にかかる管理知識が必要となる．

自律的な障害管理に関する既存の KNMS では，エキスパートな管理者が障害解決を行う際の考え方や作業の手続きを，ルールや事例といった形式で蓄積し，管理知識としていた．これにより，エキスパートな管理者のふるまいに関する管理知識に基づき，KNMS がエキスパートな管理者と同様にふるまい，自律的な障害解決を図ることを可能としていた．しかし，このような KNMS が十分な自律性を発揮するためには，知識ベース内の管理知識の整合性が十分にとれていることが前提である．例えば，知識ベース内にある既存の知識と矛盾する，あるいは重複によるコンフリクトが発生するような，誤った管理知識を知識ベースに追加した場合，知識ベース内の整合性が取れなくなるため，KNMS が動作を停止することや，誤った障害原因や対策案を導出することが考えられる．また，適切でない対策案をネットワークシステムに適用した場合，障害をさらに悪化させることも考えられる．

このように，知識ベースの管理知識の整合性維持に関する問題は，KNMS の自律性に大きく影響する．ゆえに，管理者が知識ベースの更新，すなわち，知識ベースに新たな管理知識を追加したり，既存の管理知識に修正を加える場合には，知識ベース内の知識構成を熟知していなければならない．サービス構築後に構成が変化しないネットワークシステムであれば，サービス構築時に整備した知識ベー

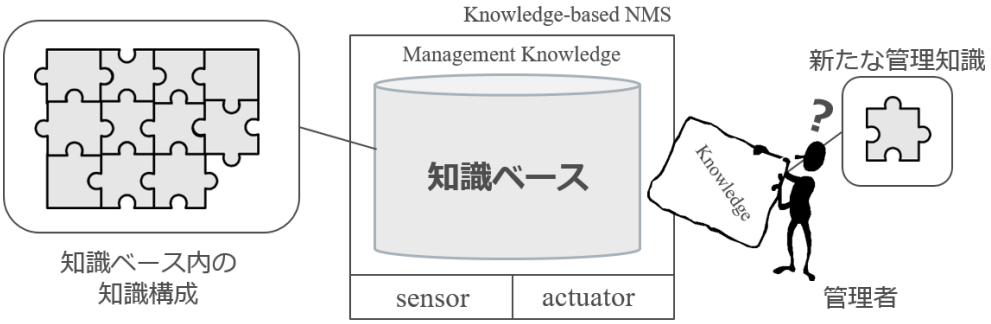


図 2.2: 自律的な障害管理のための KNMS に関する課題

スを，サービス終了まで大きな修正を加えること無く活用し続けることができるが，本研究で焦点を当てる，変動するネットワークシステムにおいては，ネットワークシステムの構成が変化する度に知識ベースの更新を行う必要があるため，知識ベースの更新にかかる作業負担や心的負担は無視できない．よって，**（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難**という課題は，変動するネットワークシステムへの適用が可能な KNMS を実現する上で解決する必要がある （図 2.2）.

2.1.3　自律的な性能管理に関する取り組み

性能管理の自律化に関して，クラウド型システムの制御をはじめとした様々な取り組みが行われている．ネットワークシステムの性能管理に関する管理者の主なタスクは，安定的に ICT サービスを提供するため，変動し続けるサービス利用者の需要に合わせて，サービスに与える計算資源の量を調整したり，サービスを構成するソフトウェアの設定を変更したりすることである．この時，適切に ICT サービスの制御を行うためには，運用中のネットワークシステムの状態を的確に見極めることが重要である.

図 2.3 に示すグラフは，ネットワークシステムの活動度（x）に対する処理量（μ）によってシステムの処理能力を示した，スループット（T）と呼ばれるシステム特

性であり，理論的には T_{ideal} のように単調増加飽和型の形状をもつ．しかし，実際の運用では，T_{real} のような釣り鐘型の形状となり，処理量が最大となる限界状態 m を超えた過負荷状態では，たとえ活動度が増加しても処理量が減少する状態に陥る．一般に，稼働中のシステムが T_{real} 上のどの状態で動作しているのかを特定することは容易ではない．そのため，例えば，T_{real} 上の状態 a 付近で動作していたシステムが限界状態 m に近づいたこと，あるいは，システムが限界状態 m を超えた過負荷状態 b で動作していることを判定することはできない．そこで，ネットワークシステムの構築時に，あらかじめ所定の状況におけるスループット特性を把握しておき，システム運用時にこの特性，および要求量や処理量の実測値を利用して，システムの運用状態の推測が行われる．例えば，システムが状態 a から状態 m に推移しつつあると判断された場合には，システムの活動度を維持するために処理量の調整やスケーリング操作が施される．ところが，こうしたスケーリング操作によってシステムの機能や構成が変化すると，図 2.4 に示すように，スループット特性も同時に変化する．例えば，スループット特性が $T(t_0)$ から $T(t_1)$ に変化した場合，システム性能の限界点も状態 m から状態 b に変化する．しかし，変化後の特性 $T(t_1)$ が把握できない場合には，システムの動作状態の判定はきわめて難しくなり，システム運用にも支障が生じる．例えば，システムが限界状態 x_2 を超えた状態にあることが判断できない場合，システムは望ましくない状態に陥ってしまう．

上述した性質をもつネットワークシステムの性能管理を行うため，システムの運用や制御に関する種々の手法が工夫されてきた．そして，これらの手法に共通した一つの重要な問題は，不規則に到来するサービス処理要求に伴って刻々と変化するシステムの運用状態を的確に捉えるための効果的な方法を与えることである．

この問題に対して，現在の運用・管理の現場では，計算資源の使用率に基づいてシステムの制御を行うタイミングを判断する手法が広く用いられている．これ

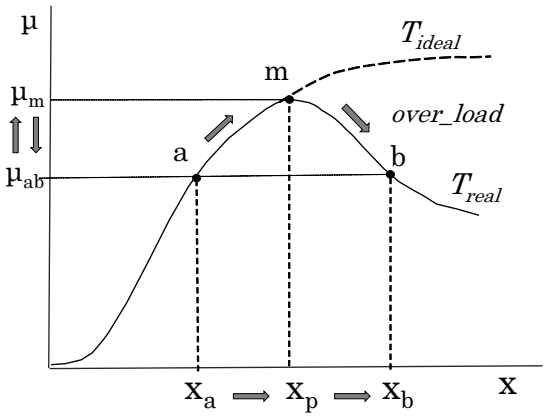


図 2.3: ネットワークシステムの特性

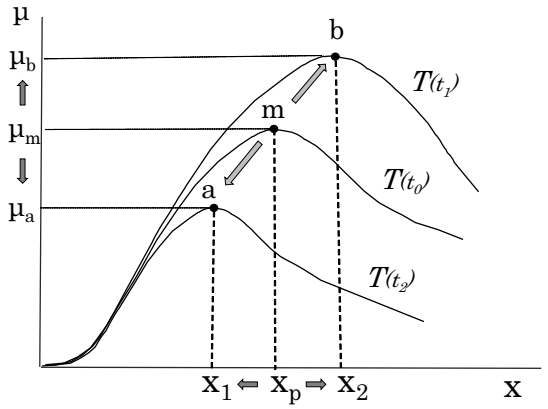


図 2.4: ネットワークシステムの特性の変化

は，ICT サービスに割り当てた CPU やメモリ，ストレージなどの計算資源の使用率を監視し，あらかじめ設定した閾値を超えた場合にスケーリングなどの処置を施す手法である [21, 22, 23]．これらの手法では，計算資源の使用率が高まった際にスケーリングを行うことでサービスが過負荷状態に陥ることを回避できる．一方で，仮想化環境における正確な計算資源の使用率の計測の難しさも指摘されている [24]．

また，ネットワークシステムが提供する ICT サービスの質に基づいてシステムの運用状態や性能を示す指標，すなわち QoS（Quality of Service）を考慮した性能管理に関する研究も進められている [25, 26, 27, 28, 29, 30]．実際のサービスの運

用状態を考慮することで，SLA（Service Level Agreement）や SLO（Service Level Objective）といった具体的な運用目標を性能管理の指針として取り入れることが可能となる．QoS を考慮した性能管理を行う場合，サービスに割り当てる計算資源の量とサービス需要の大きさに対する QoS の関係，すなわちネットワークシステムの動作特性を事前に把握しておく必要がある．そのため，既存の手法では，あらかじめベンチマーク [31, 32] や理論解析 [33, 34, 35, 36] などにより，管理対象とするネットワークシステムの動作特性を要求する点で，実用性の課題が指摘されている [25].

こうした QoS は，稼働中のシステムの状態を把握するための手段として，様々なシステムで広く導入されている．その一方で，実際のネットワークシステムの運用においては，高負荷時のシステムが非線形的な動態を有し，QoS が破局的に悪化する現象が生じることが知られており，サーバシステムの性能管理に関する実験でも確認されている [37, 38, 39]．よって，ネットワークシステムの動作特性，特に高負荷時のシステムのふるまいは，実際にシステムに大きな負荷を与え，過負荷状態を引き起こしてみなければ，知り得ない情報である．ゆえに，本研究で焦点を当てる変動するネットワークシステムにおいては，構成が変化する度にシステムの動作特性を計測し直す必要があるが，実運用上困難であることが分かる．以上より，**（P2）動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難**という課題は，変動するネットワークシステムへの適用が可能な KNMS を実現する上で解決する必要がある（図 2.5）.

2.2　研究課題

前節にて，知識型ネットワーク管理システムに関する取り組みに関して，特に自律的な障害管理と性能管理に関する取り組みに焦点を当て，変動するネットワークシステムへの適用時に発生する課題を詳細化した．自律的な障害管理に関する

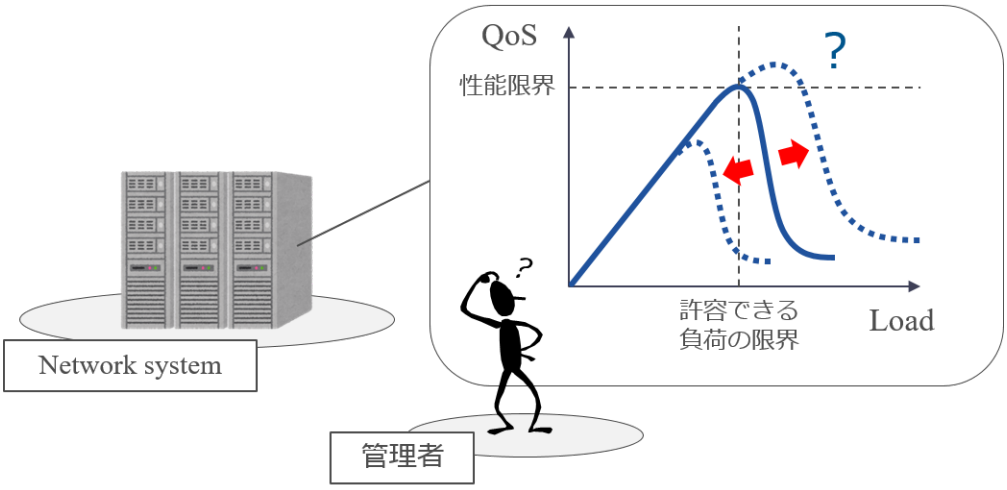


図 2.5: 自律的な性能管理のための KNMS に関する課題

KNMS には，**（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難**，という課題があり，そして，自律的な性能管理に関する KNMS には，**（P2）動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難**，という課題があった．管理対象とするネットワークシステムの構成変化に伴って知識ベースを更新する際，知識の整合性管理にかかる負担が大きく，容易に管理知識の追加や変更ができないこと，そして，システムの詳細な動作特性に関する管理知識は実運用上獲得・更新が困難であることにより，既存の KNMS に関する取り組みは知識ベース更新の必要の無い，変動しないネットワークシステム，あるいは，どのように構成が変化するかをあらかじめ把握可能なネットワークシステムに適用範囲が限定されてきたと言える．よって，（P1）および（P2）は，変動するネットワークシステムへの適用が可能な KNMS を実現する上で解決すべき課題である．

2.3　課題解決のアプローチ

前節にて述べた既存の KNMS に関する課題について，本研究における解決のアプローチを以下に述べる．

（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難 という課題については，**（S1）モジュール化に基づくサービス指向型の知識管理法** を提案する．本アプローチでは，管理知識をモジュール化し，モジュールの集合として知識ベースを表現する．これにより，新たに管理知識を追加・変更する場合に，知識ベース全体の知識構成を把握していなくても，管理知識を含むモジュールの追加・変更により，システム構成の変化の原因となった ICT サービスに着目した，部分的な知識ベースの更新を容易に行うことを可能とする．本提案により，知識ベースを更新する際の知識の整合性確認に要する管理者負担を軽減し，変動するネットワークシステムにおいても自律的な障害管理が可能な KNMS の実現を目指す.（S1）については，3 章にて詳説する．

（P2）動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 という課題については，**（S2）巨視的な観点に基づくネットワークシステムの内部状態の観測法** を提案する．本アプローチでは巨視的な観点からのネットワークシステムの分析，すなわち，ネットワークシステム全体としての挙動を捉え，分析する．これにより，変動するネットワークシステムにおいて頻繁に変化する，そのシステムの動作特性に依存せずに，ネットワークシステムの動作状態を的確に捉えることができる，汎用的な管理知識の記述を可能とする．本提案により，実運用上獲得することが難しい詳細な動作特性に関する管理知識を用いずとも，運用中のネットワークシステムの状態を捉えることを可能とし，変動するネットワークシステムにおいても自律的な性能管理が可能な KNMS の実現を目指す．なお,（S2）については，4 章にて詳説する．

第3章　知識型ネットワーク管理システムによる自律的な障害管理

第 3 章では，知識型ネットワーク管理システムを用いた障害管理タスクの自律化に焦点を当て，熟練管理者の専門知識の蓄積と利用を支える知識ベースの構築法について検証し，知識のモジュール化を導入したサービス指向型の知識管理手法を提案する．

3.1　概要

近年，ICT サービスを提供するための環境構築に際して，仮想化技術が広く用いられている．従前の手法では，計算資源とオペレーティングシステムの関係が固定的であった．このような手法で構築された ICT サービスは，システムの構成が明確であるため，管理を直感的に行うことができる利点があるが，柔軟性に乏しいという欠点がある．その一方，仮想化技術に基づくサービス構築手法では，ハイパーバイザや仮想化ソフト上に仮想マシンを作成・配置することで，迅速に ICT サービスを構築するとともに柔軟に構成を変化させることができる．そのため，仮想化された計算資源上に構築された ICT サービスは，利用者需要の変動に合わせて柔軟に対応することが可能である．また，ICT サービスに対する利用者需要の変動は以下に示す二種類がある．

利用者需要の量的な変動

ICT サービスに対する利用者需要の量的な変動は，すなわち需要の大きさの変化である．利用者需要の量的な変動に対しては ICT サービスに与える計

算資源の量を変化させることで柔軟に対処することが可能である．例えば，仮想マシンのサイズ変更や，複製した仮想マシンの着脱により，計算資源の量を調整することができる．

利用者需要の質的な変動

ICT サービスに対する利用者需要の質的な変動は，すなわち需要の種類の変化である．利用者需要の質的な変化に対しては ICT サービスの構築フェーズと運用フェーズを短期間で繰り返すことで対処することが可能である．仮想化された計算資源に基づく ICT サービスはハードウェア面での制約が少ないため，十分に設計を固めてからサービスを構築するウォーターフォール型の開発手法でなく，試作を積み重ねながらサービスの質を徐々に高めていくような開発手法をとることができる．また，このような仮想化技術の利点を活用することで，大規模災害時のサービス復旧に関する取り組みも行われている [40, 41]．

このように，仮想化技術の進展により ICT サービスの柔軟性が向上する一方で，ネットワーク管理者は利用者需要の変動に伴って発生するネットワークシステムの構成変化に対応する必要がある．特に，利用者需要の質的な変化によりサービスを提供するための多種多様な機能が短期間の内に導入・削除される場合，そのようなネットワークシステムの障害管理に携わる管理者には，さらに広い知識と豊富な経験が要求される．

2 章にて述べたとおり，自律的な障害管理の実現に向けた KNMS に関する取り組みは，（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難 という課題により，その適用範囲を，構成の変化しない固定的なネットワークシステムに限定されてきた．本章では，この課題に対して（S1）モジュール化に基づくサービス指向型の知識管理法 を提案することで，知識ベースを更新する際の知識の整合性確認に要する管理者負担を軽減し，変動するネット

ワークシステムにおいても自律的な障害管理が可能な KNMS の実現を目指す．また，提案に基づいて試作システムを設計・実装し，評価実験を行うことで本提案の有効性を示す．

3.2　関連研究と課題

ネットワーク管理者の障害管理を支援するため，これまで NMS について様々な研究開発が行われてきた．近年，ネットワーク管理で一般的に使用されている NMS は，SNMP などの管理用プロトコルを用いて機器の状態を監視する機能を中心として設計されている．このような NMS は，管理対象の正常状態に関する判断基準を設定することで，管理者に代わって管理対象システムの異常を検知することができる．しかし，詳細な障害原因や障害発生場所の特定，対策案の策定などは管理者自らが行う必要があり，障害管理タスクのなかでも部分的な自動化にとどまっている．

このような観点から，人間の管理者が持つ高度な知識を NMS に付与することでその能力を強化する，KNMS に関する取り組みが行われている．2 章にて述べたとおり，障害管理の自律化に関して，以前よりエキスパートシステムをはじめとした様々な取り組みが行われてきた．自律的な障害管理を目指す KNMS は，ネットワークシステムに障害が発生した場合に，管理知識に基づいて障害の発生箇所や原因を特定するとともに，適切な対応策を導出し，適用するための機能を有する．

しかしながら，KNMS を有効活用するためには，ルールや事例といった形式で記述された管理知識を，管理対象システムの構成に合わせて絶えずメンテナンスし，その整合性を維持し続ける必要がある．システム構成に適さない知識は KNMS の誤動作やパフォーマンスの低下を引き起こす原因となるため，管理者は知識ベース内の知識構成を熟知し，知識の整合性に細心の注意を払わなければならない．従来の KNMS に関する試みは，主に管理対象システムの構成が変化しない固定的な

環境を対象としていたため，サービスの運用開始後における知識の追加や変更に伴う管理者負担を考慮する必要性は低かった．その一方で，本論文で議論する，変動するネットワークシステムに対して KNMS を適用する場合，前述のとおり，利用者需要の変動に伴ってシステム構成が頻繁に変化するため，その変化の都度，知識ベースに知識を追加・変更しなければならず，知識ベースの更新にかかる作業負担や心的負担は無視できない．よって，**（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難** という課題は，変動するネットワークシステムに適用可能な KNMS の実現に向けて解決する必要がある．

3.3　モジュール化に基づくサービス指向型の知識管理法

3.3.1　提案の概要

上述の課題を解決するため，本研究では，**（S1）モジュール化に基づくサービス指向型の知識管理法** を提案する．図 3.1 に，本提案による課題解決のアプローチを示す．本アプローチでは，ネットワークシステムの構成を，ICT サービスを提供するための機能要素の集合として捉え，管理知識を任意のまとまりでモジュール化する．本提案のねらいは，これまでモノリシックな造りであった知識ベースを，ネットワークシステムを構成する ICT サービス毎に任意のまとまりで分割し，個別に管理可能とすることで，その管理性の向上を図ることにある．管理知識を含むモジュールの追加・変更により，システム構成の変化の原因となった ICT サービスに着目した，部分的な知識ベースの更新ができる．ゆえに，本提案に基づく KNMS では，知識ベース内の知識構成を完全に把握していない管理者であっても，新たに管理知識の追加や変更が可能となる．

本提案は知識ベースのモジュール化，すなわち，知識ベースを分割し個別に管理可能とするものである．いわば，本提案に基づくシステムは，複数の知識ベースから成る KNMS となるが，複数の知識ベースを扱うことに関して管理者負担が

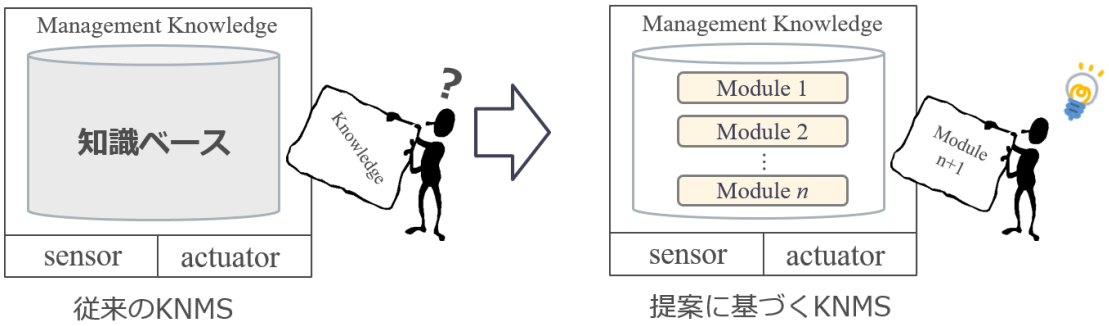


図 3.1: 課題 (P1) の解決に向けたアプローチ

増大することが懸念される．そこで本研究では，この問題を解決するために，ソフトウェアエージェントのもつ自律性・社会性に着目し，先行研究であるエージェント型の KNMS を拡張することで，提案の実現を目指す．

3.3.2　先行研究

本項では，先行研究である，能動的情報資源に基づくネットワーク管理システム（Active Information Resource-Based Network Management System: AIR-NMS）について説明する．

能動的情報資源（Active Information Resource: AIR）は，分散環境上に存在する電子的なデータに対して，そのデータを活用するための知識（利用支援知識）と機能（利用支援機能）を付与し，自律的に動作可能なエージェントとする考え方である [42]．利用支援知識には，情報資源の活用に関する知識や他の AIR との連携に関する知識が含まれる．また，利用支援機能には，情報資源の活用にかかる処理を行う機能や，他の AIR と連携するための通信機能が含まれる．AIR の概念を用いることで情報資源の活用時に必要な，情報の検索・取得・加工といった作業を AIR が自律的に代行し，利用者の負担を軽減できる．

AIR-NMS は AIR の概念をネットワークシステムの管理に適用したものである [20]．AIR-NMS は主に以下に示す 2 種類の AIR 群によって構成される．

Knowledge-AIR（K-AIR）

ネットワークシステムの管理に関する知識を情報資源と捉え，それに利用支援知識と利用支援機能を付与したもの．

Information-AIR（I-AIR）

ネットワークシステムを構成するハードウェア・ソフトウェアの状態情報を情報資源と捉え，それに利用支援知識と利用支援機能を付与したもの．

AIR-NMS は K-AIR 群と I-AIR 群，すなわち，エージェント化された管理知識とネットワークシステムの状態情報が，自律的にメッセージを交換しながら互いに協調連携することで人間の管理者の作業を代行し，管理者の作業負担を軽減するものである．

AIR-NMS の概念の応用として，これまで障害管理に焦点を当てた取り組みが行われている [20, 42, 43, 44]．図 3.2 に，障害解決における AIR-NMS を用いた管理者支援の概要を示す．

K-AIR は上級管理者や管理マニュアル等から得られる障害対応に関する管理知識を記述したファイルを情報資源とする．K-AIR には，利用支援知識として，関連する管理知識をもつ K-AIR や障害原因の特定と対策案導出に必要な機器情報を持つ I-AIR を探索し，相互に連携して障害解決を行う知識を与える．また，利用支援機能として，ファイルに記述された管理知識を AIR に解釈可能な形式で抽出する機能を与える．

I-AIR には利用支援知識として，管理対象機器の監視設定に関する知識や，K-AIR からの情報要求に応答可能かを判断する知識を与える．そして，利用支援機能として，管理対象機器の動作状況を取得，加工し，蓄積する機能を与える．

AIR-NMS による管理者支援の具体的な流れを以下に示す．

1. はじめに，運用中のサービスに異変を感じた管理者がユーザインタフェース

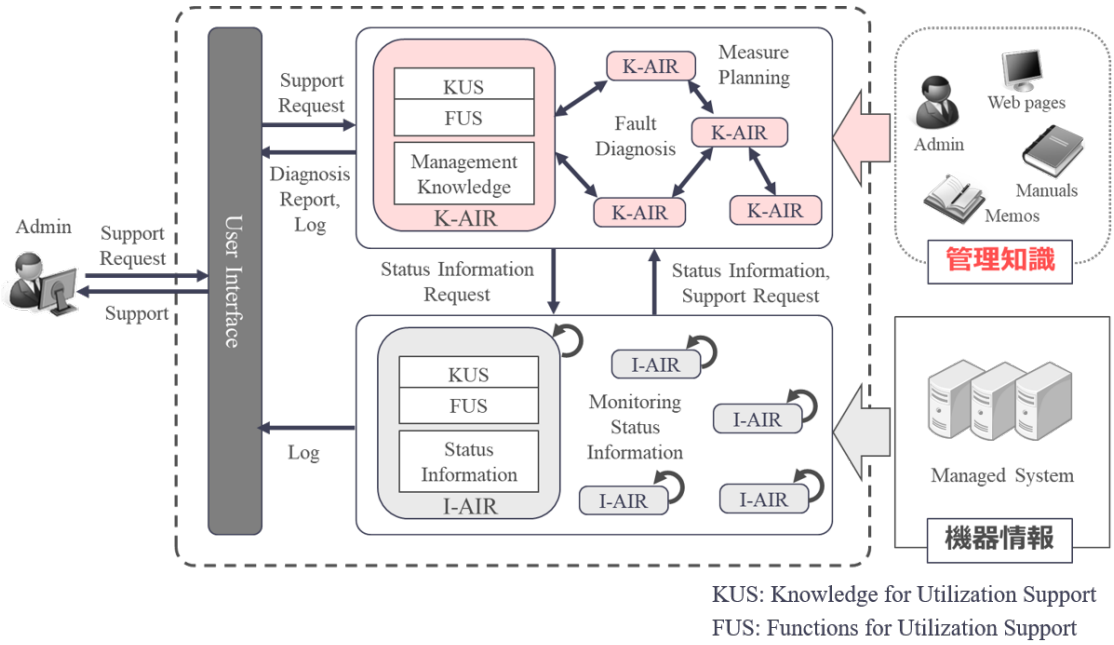


図 3.2: 能動的情報資源に基づくネットワーク管理システム（AIR-NMS）

を介して AIR-NMS に障害解決支援要求を送る，あるいは，管理対象機器の異変を検出した I-AIR が K-AIR 群に対して障害解決支援要求を送ることにより，障害解決支援が開始される．

2. 要求を受け取った K-AIR が他の K-AIR と連携することで，障害原因の推論を行う．この際，推論をすすめるために管理対象機器の情報が必要な場合には，適宜 I-AIR 群に問い合わせることで機器情報を取得する．

3. 推論の結果として障害原因が特定された場合には，特定された障害原因に対応する対策案を生成し，管理者に提示する．

4. 管理者が提示された対策案を実行する，あるいは，I-AIR が生成された対策案を管理対象機器に適用することで，ICT サービスを復旧することができる．

従来の AIR-NMS に関する取り組みもまた，ネットワーク運用の自律化を目指す KNMS の取り組みの一つである．AIR-NMS における知識ベースは K-AIR 群が動作する場所であるが，これまでの AIR-NMS に関する取り組みでは，知識ベー

ス内の K-AIR は自由にメッセージを交換し，連携することが可能であった．そのため，新たに管理知識（K-AIR）を知識ベースに追加する場合には，それまでに知識ベースに蓄積された K-AIR との連携，エージェント間メッセージの送受信について考慮する必要がある．この作業にあたり管理者は知識ベース内の K-AIR 群がどのようにメッセージを送受信し連携するか，すなわち，知識ベース内の知識構成を熟知していなければならない．よって，これまでの AIR-NMS に関する取り組みもまた，変動するネットワークシステムへの適用に関する課題（P1）が同様に当てはまる．

これまで，AIR-NMS はマルチエージェントシステムとして設計・実装されてきた．マルチエージェントシステムとして設計された AIR-NMS は，システム全体としての能力としてエージェントの連携によってタスクを完遂すると同時に，システムを構成する各エージェントも，単体で完結した動作をすることができる．AIR-NMS では，いわば知識ベースがエージェント化された管理知識（K-AIR）の集合として表現されるため，知識ベースを複数に分割したとしても，K-AIR の持つ能動性により，分割された単位で有効に機能しタスクを実行することができる．さらに AIR-NMS に関する先行研究として，K-AIR について，複雑な管理知識を，簡単で小さな管理知識の集合として表現するような設計事例があり，これは，ネットワークシステム全体に関する管理知識を，サービス毎に分割された管理知識の集合として表現する本提案の実現に有用である．

本研究では，以上に示すとおり，AIR-NMS が知識ベースを分割しやすい設計であること，そして，知識ベースの分割に際して新たな管理者負担が発生しない点に着目し，これらの点が知識ベースのモジュール化に有用であると考え，AIR-NMS に関するこれまでの取り組みを拡張することで提案の実現を目指す．

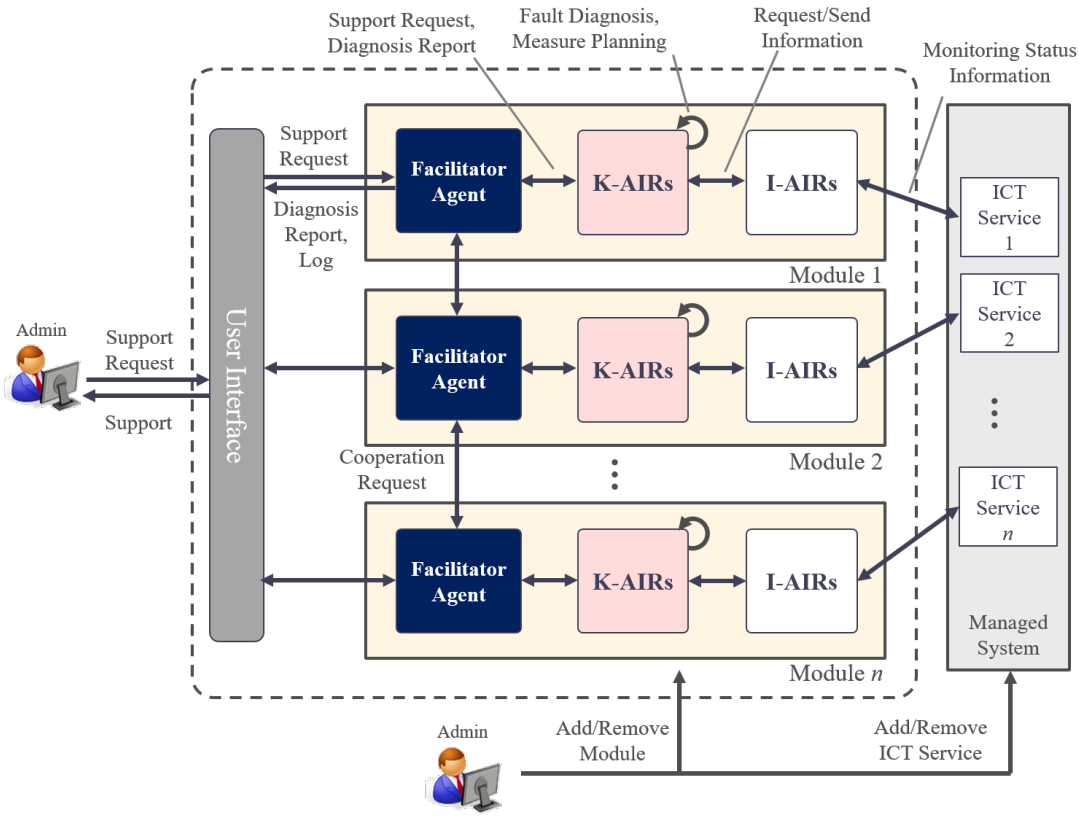


図 3.3: 提案システムのアーキテクチャ

3.3.3　サービス指向型の知識管理法

図 3.3 に，サービス指向型の知識管理法により拡張された AIR-NMS の概要を示す．本提案ではネットワークシステムの構成変化に応じた柔軟な知識の追加・変更を可能とするため，AIR-NMS をモジュール化する．各モジュールは管理知識である K-AIR 群，そしてネットワークシステムの状態情報である I-AIR 群，そして，後述する Facilitator-Agent から構成される．管理対象システムをサービス提供のための機能要素の集合として捉えることで，任意の機能要素のまとまりを一つのサービスとして切り分け，サービス毎に独立したモジュールとして管理知識を与える．ネットワークシステムに新たなサービスが追加された場合や，既存のサービスの機能に変化があった場合には，そのサービスに関するモジュールを追加・変更することでシステム構成の変化に対応する．

提案システムでは，K-AIR をサービス毎に分割して管理することで，知識の整合性を考慮すべき範囲を各サービスのモジュール内に限定する．これにより，システム構成の変化の原因となったサービスに焦点を当てた，他のサービスに関する知識と独立した知識の記述が可能となる．ゆえに，管理者は知識ベース全体の知識構成を把握していなくても，モジュールの接続および離脱により，部分的な知識ベースの更新を容易に行うことができる．

しかしながら，実際の ICT サービスの運用現場において，ネットワークシステムの構成を，機能要素が重複すること無く，それぞれ独立したサービスとして切り分けることは困難である．例えば，データベース機能や認証系の機能等は，複数のサービスに含まれる可能性がある．機能要素の重複無く各サービスを独立したものとして切り分けようとした場合，一つ一つのサービス規模が大きくなる可能性があり，その管理知識を含むモジュールも肥大化してしまう．この場合，知識の整合性確認にかかる管理者負担が小さいというモジュール化のメリットが失われ，部分的な知識ベースの更新が困難となる．そこで本提案では，サービス間で機能要素の重複，すなわち，サービス間の依存関係を許容しつつもモジュール化のメリットを維持するため，新たに Facilitator-Agent を導入する．

Facilitator-Agent

管理知識の管理性を考慮しモジュールを小さく保つには，サービス間の依存関係を許容する必要があるが，他のサービスと依存関係のあるサービスに不具合が発生した場合，複数のモジュール間で管理知識や機器情報を組み合わせなければ十分な障害解決支援が困難となる．本提案ではモジュール化の利点を損なわず，かつ，必要に応じてモジュール間で K-AIR を連携させた障害解決支援を可能とするために，Facilitator-Agent を導入する．図 3.4 に，Facilitator-Agent を介した，モジュール間の管理知識連携の概要を示す．Facilitator-Agent はいわばモジュール

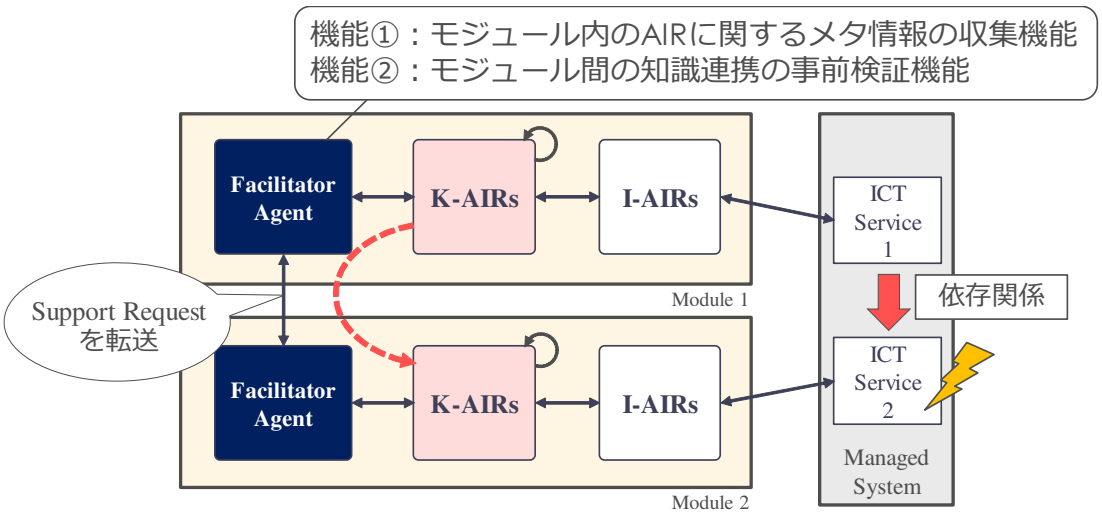


図 3.4: Facilitator-Agent を介したモジュール間の管理知識連携

の窓口であり，モジュール間で管理知識の連携を行う必要がある場合には，全て Facilitator-Agent を介して連携する．Facilitator-Agent はモジュール化の利点を保つため，以下の機能を持つ．

1. モジュール内の AIR に関するメタ情報の収集機能

   モジュール内の K-AIR や I-AIR に対して，それぞれの AIR がどのような管理知識や機器情報を保有しているか問い合わせる機能である．これにより，モジュール内にどのような種類の管理知識や機器情報が存在するかをリスト化する．

2. モジュール間の知識連携の事前検証機能

   他のモジュールから送信された障害解決支援要求のメッセージに対して，当該モジュールで対応可能であるかどうか，すなわち，必要な管理知識と機器情報がモジュール内に存在するかどうかを，前述の機能で作成したリストを参照することで検証する機能である．当該モジュールで対応可能であると判断された場合には，障害解決支援のメッセージを受理し，モジュール内の K-AIR にメッセージを転送する．

以上に示すとおり，モジュール間で AIR が直接通信することを制限し，モジュール化の利点を維持する一方，モジュール間で AIR を連携させる必要がある場合には，Facilitator-Agent によって管理知識の連携を事前に検証した上で問題が無いと判断された場合のみ間接的に管理知識を連携させることで，依存関係のあるサービスに起因する障害解決タスクの完遂を可能とし，AIR-NMS の実用性の両立を図る．

3.4　評価実験

本提案の評価を行うため，試作システムを設計・実装し，実験を行った．

3.4.1　設計

試作システムは先行研究 [20] での設計・実装をふまえ，DASH/IDEA エージェントフレームワーク [45, 46] を実装基板と想定して設計を行った．

K-AIR の設計

AIR-NMS による障害解決支援の先行研究と同様に，試作システムにおいても，障害解決の過程を 3 つに分割した，以下の 3 種類の K-AIR を導入する．

- Ksc-AIR: 発生した障害症状（symptom）から推測される障害原因（cause）の候補に関する知識（Ksc）を扱う AIR
- Kcd-AIR: 障害症状（cause）を特定するための具体的な診断手法（diagnosis）に関する知識（Kcd）を扱う AIR
- Kcm-AIR: 特定された障害原因（cause）を解決するための対策案（measure）に関する知識（Kcm）を扱う AIR

```
<sc symptom="SIP client cannot make calls">
   <info>SIP-client_ID</info>
   <cause>SIP server process down</cause>
   <cause>SIP connection port not open</cause>
   <cause>Client ID not permitted to call</cause>
</sc>
```

図 3.5: Ksc の記述例

図 3.5，図 3.6，図 3.7 に各 K-AIR が保有する知識，すなわち，Ksc，Kcd，Kcm の記述例を示す．これらの知識は XML（Extensible Markup Language）形式で記述されており，記述例は SIP（Session Initiation Protocol）サービス（IP 電話サービス）の障害解決に関する知識の一部である．

図 3.5 に示す Ksc は，IP 電話が発信できないという障害症状から，cause タグで囲まれた 3 種類の障害原因を導き出す知識である．この Ksc を保有する Ksc-AIR は 3 種類の障害原因に関してそれぞれ Kcd-AIR 群に対して障害解決支援要求のメッセージを送信する．

図 3.6 に示す Kcd は，SIP サービスの接続ポートが遮断されているという障害原因が実際に管理対象のネットワークシステムで発生しているかを確認するための知識である．Kcd-AIR は，Ksc-AIR から当該の障害原因に関する障害解決支援要求のメッセージを受信した場合，知識に従って I-AIR と連携しながら機器情報を収集することで，障害原因の発生に関して真偽を判定する．ここで，障害原因の発生が真であると判定された場合には，Kcm-AIR 群に対して対策案生成要求のメッセージを送信する．

図 3.7 に示す Kcm は，SIP サービスの接続ポートが遮断されているという障害原因の解決策を導出するための知識である．Kcm は対策案生成のためのテンプレートになっており，Kcm-AIR は I-AIR と連携しながら機器情報を取得し，テンプレートの穴埋めを行うことで，管理対象システムに適用可能な，より具体的な対策案を導出する．

```
<cd cause="SIP connection port not open">
  <dm>
    <p>get #SIP-port_num# cmd #grep ^port /etc/asterisk/sip.conf
    |awk -F "=" '{print $2}'# login #SIP-server_IP#</p>
    <p>get #SIP-port_state# cmd #grep val(SIP-port_num)
    /etc/sysconfig/iptables |awk -F " " '{print $NF}'#
    login #SIP-server_IP#</p>
    <p>true (#SIP-port_state# -ne ACCEPT)</p>
  </dm>
  <dr>
    Port number #SIP-port_num# is not open at server #SIP-server_IP#.
  </dr>
</cd>
```

図 3.6: Kcd の記述例

```
<cm cause="SIP connection port not open">
  <m>
  Open UDP port #"^port="value"$"#.
  1. Login server #SIP-server_IP# as superuser.
  2. Open /etc/sysconfig/iptables and add a discription of below.
   -A INPUT -m state --state NEW -m udp -p udp --dport #"^port="value"$"#
  3. Reload iptables with "/etc/init.d/iptables restart".
  </m>
</cm>
```

図 3.7: Kcm の記述例

I-AIR の設計

ネットワーク機器・サーバからの情報取得機能など，基本機能の設計に関しては先行研究 [20] に従う．また，汎用的な情報取得機能の実装のため，機器への接続に関して，人間の管理者が機器情報を取得する際に用いる Secure Shell の利用に関する知識を利用支援知識として付与した．

Facilitator-Agent の設計

モジュール内の AIR に関するメタ情報の収集機能として，モジュール内の K-AIR が保有する管理知識や I-AIR が取得可能な機器情報に関するメタ情報を問い合わせるプロトコルを設計した．図 3.8 は K-AIR が保有する知識に関するメタ情報，すなわち，当該モジュールで解決支援が可能な障害原因名の取得例であり，取得元の AIR 名と対応づけてリスト化される．他のモジュールの Facilitator-Agent か

```
(SIP server process down,Ksc-AIR.201412111553015:w2:AIR-NMS-PC-1)
(SIP connection port not open,Kcd-AIR.201412111553020:w3:AIR-NMS-PC-1)
(Client ID not permitted to call,Kcd-AIR.201412111553023:w3:AIR-NMS-PC-1)
```

図 3.8: モジュール内の K-AIR に関するメタ情報の取得例

らの障害解決支援要求に対しては，このリストに基づいて対応可能性を判定する．

また，通信可能な他モジュールの Facilitator-Agent の検索には，DASH/IDEA のネームサーバ機能を使用し，定期的に再検索を行うことでモジュールの追加・削除に対応する．

3.4.2　実験概要

評価実験のため，設計に基づいた試作システム，すなわち，モジュール化を導入した AIR-NMS を実装した．実装およびエージェントの動作環境には，DASH/IDEA フレームワークを使用した．さらに，実験結果の比較より提案の効果を確認するため，先行研究 [20] に基づくシステムを実装した．実験では，提案に基づく試作システム（提案システム）と先行研究 [20] に基づくシステム（既存システム）をそれぞれ用いて，障害解決支援を行う．

実験 1 では，実際に障害解決の一部である障害診断を実行し，その実行結果の比較を行う．実行結果の比較については，以下の 2 点について行う．

- 出力された診断結果．特に，特定された障害原因と導出された対策案について比較する．
- 障害原因の特定と対策案の導出にかかる AIR/エージェント間メッセージ数を比較する．

実験 2 では，実装した実験システムに基づいて知識ベース更新のシミュレーションを行い，新たに管理知識を追加する際に整合性確認の必要がある管理知識の数を比較する．

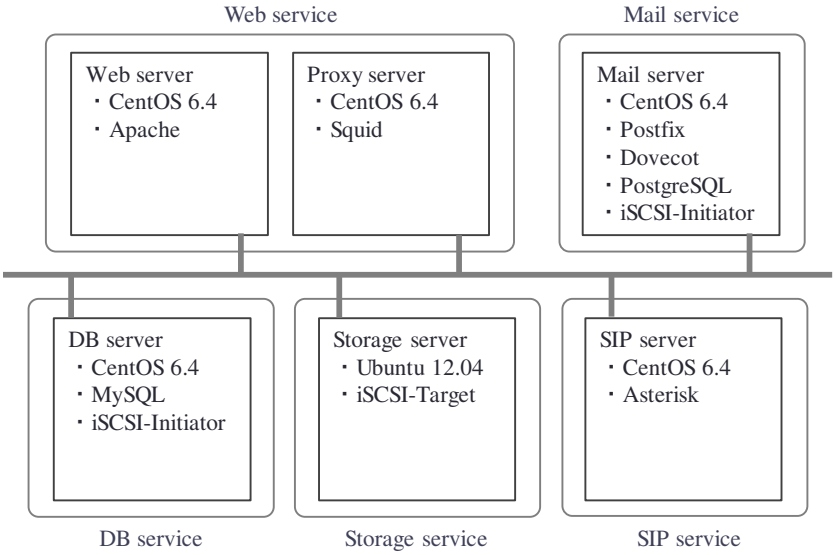


図 3.9: 実験用ネットワークシステムの構成

図 3.9 に，実験用ネットワークシステムの構成を示す．管理対象システムには，5 種類のサービスが運用中である．また，Web サービスと DB サービス，DB サービスと Storage サービス，Mail サービスと Storage サービスには依存関係がある．

実験に使用する各 ICT サービスの管理知識の個数を表 3.1 に示す．本実験では，Kcd と Kcm を一対一対応として作成した．提案システムと既存システムで与えた知識は同じであるが，提案システムでは ICT サービス毎にモジュール化して与え，既存システムでは全ての知識を集約して与える点が異なる．また，本実験では，知識ベース全体の管理知識の構成を知らずに管理知識を追加したことを想定し，知識ベース全体としての整合性は考慮せず，サービス毎の知識の整合性のみを考慮して知識を記述した．ゆえに，全ての知識を集中管理する既存システムにおいては，意図しない動作を引き起こす可能性がある．

3.4.3　実験 1：障害診断の実行

提案システムと既存システムを用いて，以下の 5 通りの障害診断を実行した．

1. DB サーバプロセスダウン（障害症状：Web ページ閲覧不可）

表 3.1: 各 ICT サービスの管理知識の個数

	Number of Management Knowledge		
	Ksc	Kcd	Kcm
Web service	2	11	11
DB service	1	5	5
Storage service	1	4	4
SIP service	1	6	6
Mail service	3	13	13
Total	8	39	39

2. HTTP サーバプロセスダウン（障害症状：Web ページ閲覧不可）

3. Proxy サーバプロセスダウン（障害症状：Web ページ閲覧不可）

4. SIP サーバプロセスダウン（障害症状：IP 電話発信不可）

5. POP/IMAP サーバプロセスダウン（障害症状：メール受信不可）

実験結果を以下に示す.

診断結果の比較

図 3.10，図 3.11 に既存システム，図 3.12，図 3.13 に提案システムによる出力の一例（障害原因：DB サーバプロセスダウン，障害症状：Web ページ閲覧不可）を示す.

既存システムでは，5 通りの障害診断のうち 4 通りは正しく障害原因を特定し対策案を生成したが，1 通り（障害原因：DB サーバプロセスダウン）の診断では正しい障害原因の特定および対策案の導出に失敗した.

図 3.10 に示す，既存システムにより出力された診断報告より，2 種類の障害原因が提示されたことが分かる．どちらも障害原因「DB プロセスダウン」に関する出力であるが，それぞれ「MySQL のプロセスダウン」と「PostgreSQL のプロセスダ

ウン」に関する出力である．そして，図 3.11 に示すとおり，障害原因「DB プロセスダウン」に関する対策案が複数提示されたことが分かる．具体的には,「MySQL のプロセスの再起動の方法」と「PostgreSQL のプロセスの再起動の方法」がそれぞれ 2 回ずつ提示された．しかしながら，実験用ネットワークシステムにおいて Web サービスで使用しているデータベースは，DB サービスの MySQL であるため，PostgreSQL のプロセスダウンを障害原因とみなし，その復旧法を提示したのは誤りである．

以上の診断結果に関して，Kcd-AIR 群の動作ログより,「DB プロセスダウン」など同名の障害原因を扱う複数の AIR が障害原因の特定のために動作したことが判った．これは，管理知識の追加時に ICT サービス間の知識の整合性を十分考慮しなかった結果，知識名の重複が発生したことで，意図しない K-AIR の連携が発生したことを示している．MySQL を運用するサーバに対して PostgreSQL に関する管理知識を適用した結果，誤った障害原因の導出につながったと言える．さらに，知識名の重複により，対策案生成要求が過剰に発生した結果，同一内容の対策案が複数生成されたことも判った．

以上の結果は，知識ベース全体の知識構成を熟知している管理者でなければ知識ベースの更新が難しいという，従来の KNMS に関する課題を裏付けるものである．知識ベース内でこのような知識の衝突が発生する場合，管理者は意図しない知識連携が発生しないよう知識の修正を行う必要がある．しかし，知識の衝突を避けるためには，知識記述がより複雑かつ冗長な表現となるため，従来の集中管理型の知識ベースでは，知識の数が増えるにつれて整合性管理にかかる管理者負担が増大し続ける．

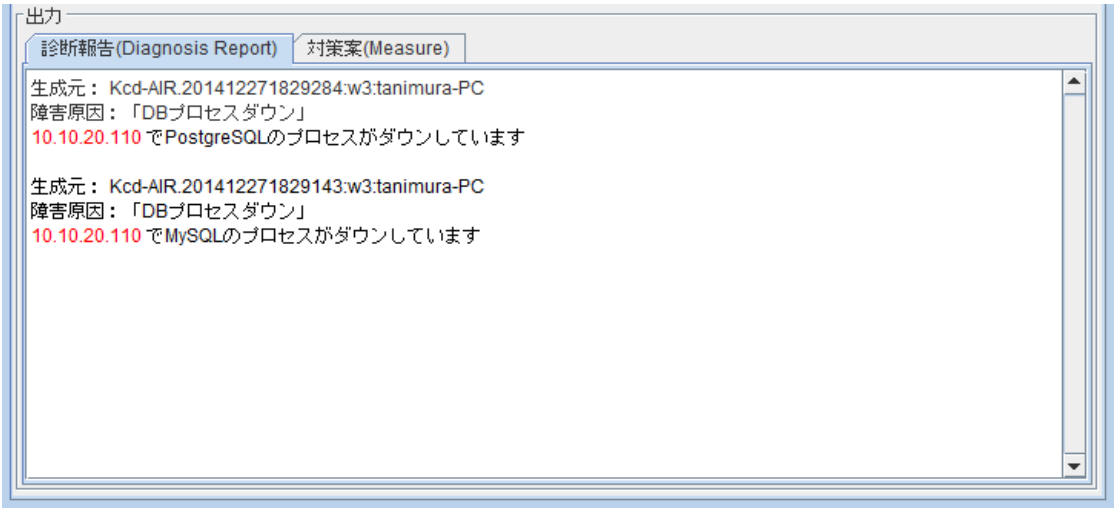


図 3.10: 出力された診断報告（既存システム）

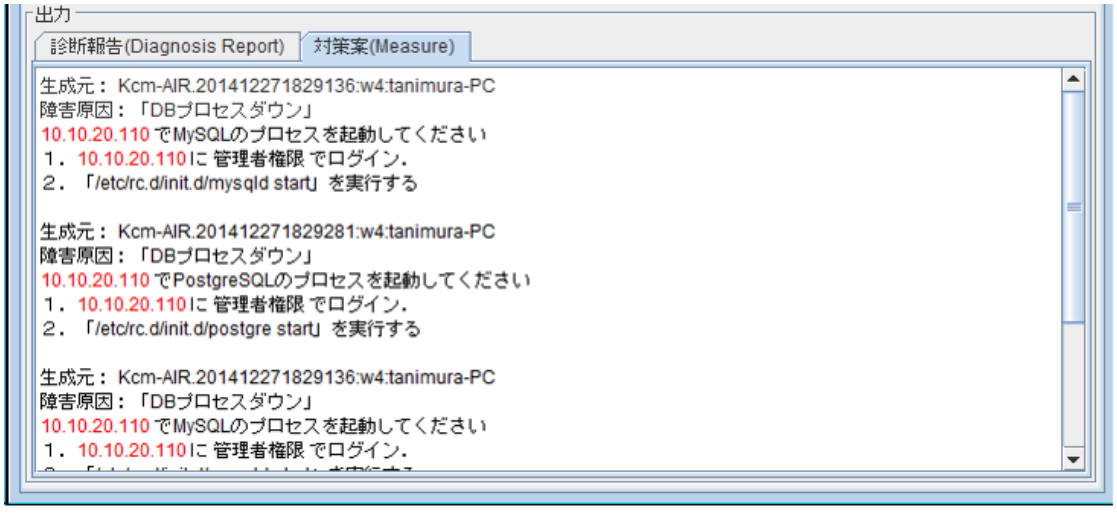


図 3.11: 出力された対策案（既存システム）

一方，提案システムでは，5 通りの障害診断全てにおいて正しく障害原因を特定し対策案を生成した．

図 3.12 に示す，提案システムにより出力された診断報告では，既存システムと異なり 1 種類の障害原因が特定され，提示されたことが分かる．そして，図 3.13 は障害原因「DB プロセスダウン」に関する診断結果であり，既存システムとは異なり対策案は 1 回のみ提示された．さらに，提示された対策案を実行することで Web サービスが復旧したことも確認した．

本診断に関して，Web サービスに関する管理知識をもつモジュールの動作ログ，および，DB サービスに関する管理知識をもつモジュールの動作ログより，Web サービスのモジュールにて開始された障害解決のプロセスが，DB サービスのモジュールに引き継がれたことが判った．具体的には，Web サービスのモジュールの Facilitator-Agent から発生した「DB サーバ接続不可」に関する障害解決支援要求のメッセージを，DB サービスのモジュールの Facilitator-Agent が受理した．そして，DB サービスのモジュールにて「DB サーバ接続不可」に関する詳細な障害原因の特定が行われ，障害原因として「DB プロセスダウン」が特定されるとともに，対策案が生成された．一方で，Web サービスおよび DB サービスのモジュールを除く他のサービスのモジュールにおいては，AIR に動作が見られなかった．これは，Web サービスのモジュールからの「DB サーバ接続不可」に関する障害解決支援要求のメッセージを，各モジュールの Facilitator-Agent がモジュール間の知識連携の事前検証機能を用いて，当該モジュールには対応可能な知識や必要な情報が不足しているとして却下したからである．

以上より，提案システムでは知識ベース全体でなくサービス毎の整合性を考慮した知識を用いた場合でも，障害解決支援が可能であることを確認した．また，サービス間に依存関係がある場合でも，Facilitator-Agent によって知識連携を検証しながら，モジュール間で知識を連携させた障害解決が可能である事を確認した．

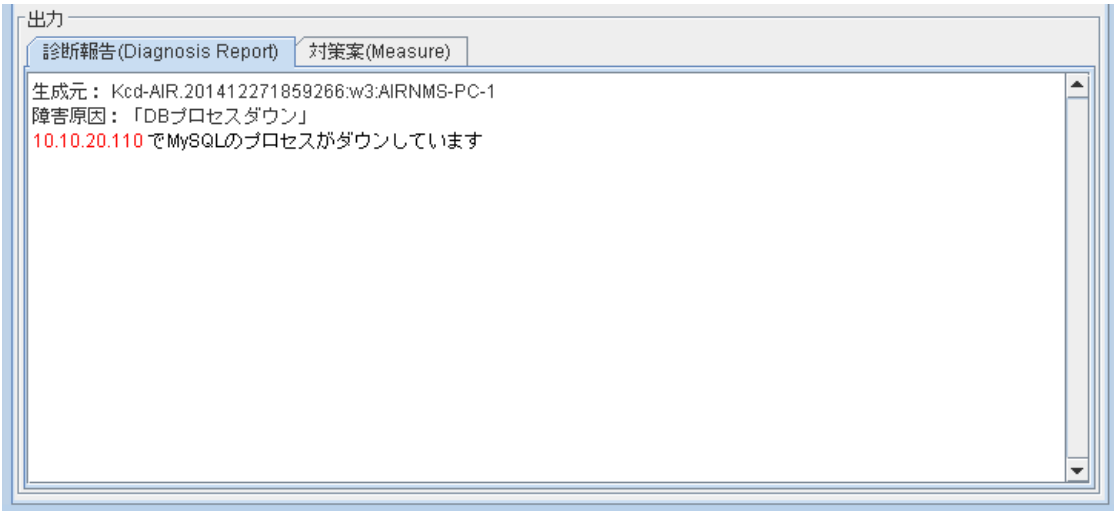


図 3.12: 出力された診断報告（提案システム）

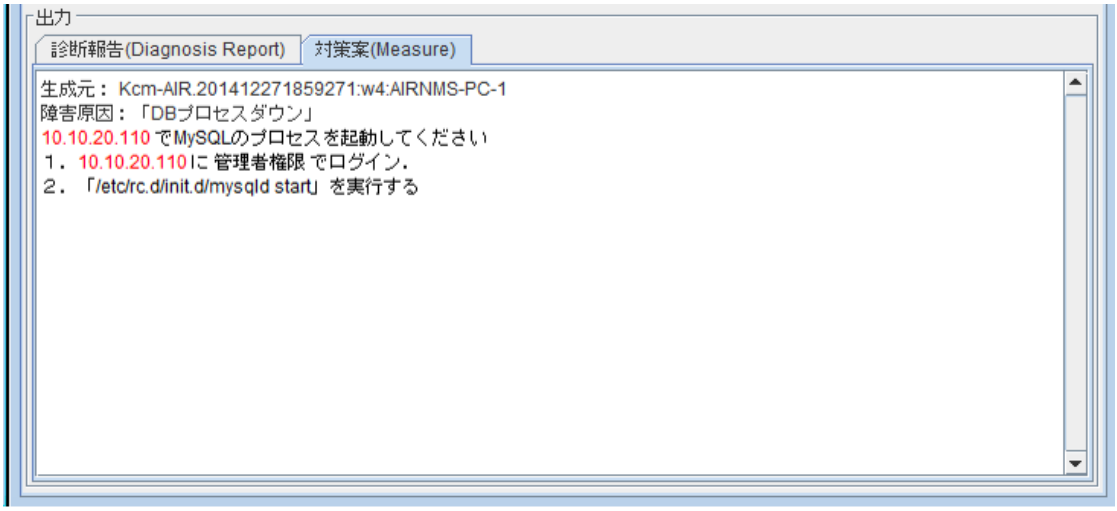


図 3.13: 出力された対策案（提案システム）

診断実行時の AIR/エージェント間メッセージ数の比較

DASH/IDEA フレームワークに基づいて実装された AIR-NMS は，メッセージ交換によって AIR およびエージェントが連携して障害解決支援を行う．本実験では，知識ベースをマルチエージェントシステムとして設計・実装したため，知識の整合性管理は，AIR およびエージェント間のメッセージの送受信を制御することと等しい．そこで，提案システムおよび既存システムの動作時に発生する AIR/エージェント間メッセージ数を計測し，比較することで，間接的に知識の整合性管理に伴う管理者負担を比較できる．

図 3.14 に，各障害原因の診断実行時に発生した AIR/エージェント間メッセージ数のグラフを示す．このグラフより，全ての障害原因の診断において，提案システムでは既存システムと比較してメッセージ数が大幅に減少していることが確認できる．特に，知識名の衝突が発生する Web サービスと Mail サービスに関する障害診断において既存システムではメッセージ数が増大する傾向にあるが，他のサービスとの依存関係の無い SIP サービスに関する障害診断においても，提案システムでは大きくメッセージ数が減少している．これは，ICT サービス毎に管理知識をモジュール化したことにより，障害解決支援要求のメッセージや情報要求のメッセージのブロードキャストドメインが，各モジュール内に限定されたことによる効果であると考えられる．この結果より，管理知識をサービス毎にモジュール化した提案システムにおいては，制御すべきメッセージの数が大きく減少することが確認された．ゆえに，提案システムは，既存システムと比較して知識の整合性管理に関する負担を軽減することが可能である．

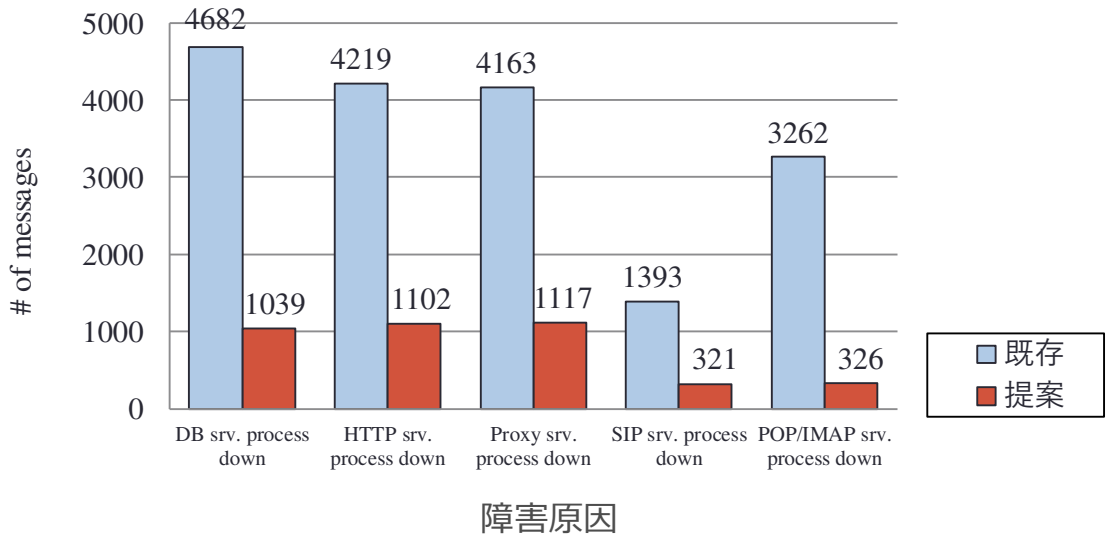


図 3.14: 診断実行時の AIR/エージェント間メッセージ数の比較

3.4.4　実験 2：知識ベース更新のシミュレーション

ICT サービスの追加や変更に伴って発生する，知識ベースへの知識の追加や変更にかかる管理者負担を比較するため，シミュレーション実験を行った．このシミュレーションでは知識ベース更新にかかる管理者負担の大きさを，新たな管理知識を追加するに当たって整合性を確認しなければならない知識の個数として表現する．具体的には，新たに追加する K-AIR 群，そして，既に知識ベース内に存在する K-AIR のうち，追加する K-AIR 群のいずれかとメッセージを送受信する可能性のある K-AIR の数である．

ある ICT サービス i に関する管理知識について，Ksc の個数を n_{sc_i}，Kcd の個数を n_{cd_i}，Kcm の個数を n_{cm_i} とすると，k 個目（$k = 1, 2, \cdots$）の ICT サービスを追加する場合の既存システムにおける管理者負担 N_k，提案システムにおける管理者負担 N'_k は以下のように表すことができる．

$$N_k = \sum_{i=1}^{k} n_{\mathrm{sc}_i} + \sum_{i=1}^{k} n_{\mathrm{cd}_i} + \sum_{i=1}^{k} n_{\mathrm{cm}_i} \tag{3.1}$$

$$N'_k = \sum_{i=1}^{k} n_{\mathrm{sc}_i} + n_{\mathrm{cd}_k} + n_{\mathrm{cm}_k} \tag{3.2}$$

（3.1）式に示すとおり，既存システムでは新たに管理知識の追加を行う場合に，他のサービスに関する管理知識を含む，全ての Ksc，Kcd，Kcm との間で整合性を確認する必要があるため，累積的に管理者負担が増大する．

一方，提案システムでは管理知識がサービス毎にモジュール化されているため，Ksc に関してはサービス間の機能要素の共有を考慮してモジュール間で整合性を確認する必要があるものの，Kcd および Kcm については他のサービスに関する管理知識との整合性確認は不要であるため，（3.2）式に示すとおりとなる．

ここで，実験 1 で使用した知識の個数（表 3.1）の平均より，各 ICT サービスの管理知識の個数について，$n_{\mathrm{sc}_i} = 2$，$n_{\mathrm{cd}_i} = 8$，$n_{\mathrm{cm}_i} = 8$ とした場合，（3.1）式，

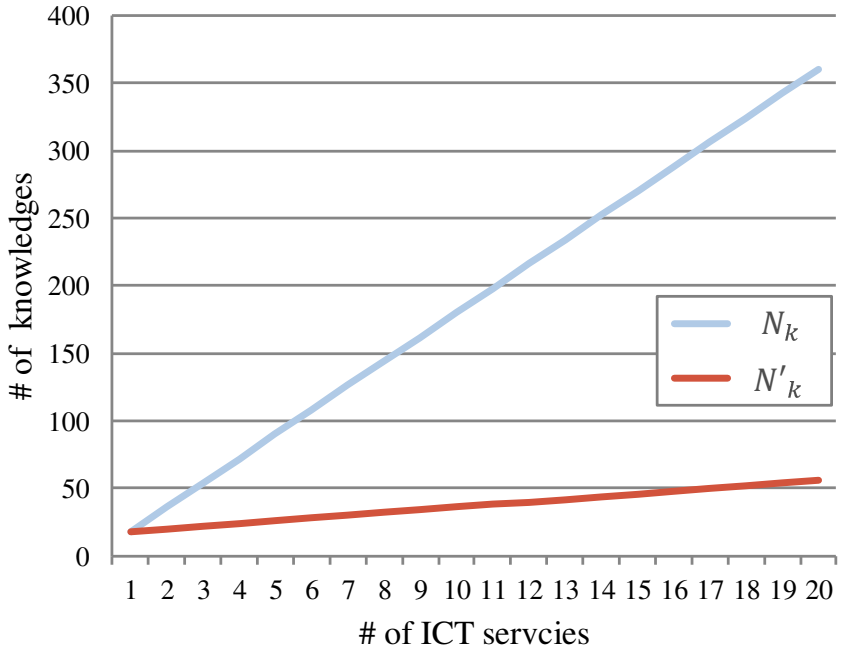


図 3.15: 知識の整合性管理に伴う管理者負担の比較

（3.2）式は以下のようになる．

$$N_k = 18k \tag{3.3}$$

$$N'_k = 2k + 16 \tag{3.4}$$

図 3.15 に，（3.3）式，（3.4）式のグラフを示す．グラフより，提案システムでは ICT サービスの数が増加した場合でも，既存システムと比較して大きく管理者負担が抑制されていることが分かる．また，AIR-NMS は抽象的な障害症状から詳細な障害原因および対策案を導出するような知識設計となっており，多くの場合，$n_{\mathrm{sc}_i} < n_{\mathrm{cd}_i}$，$n_{\mathrm{sc}_i} < n_{\mathrm{cm}_i}$ の関係である．そのため，Kcd および Kcm の整合性確認が不要となったことで，モジュール化の効果が大きく現れていると考えられる．

3.4.5　評価

実験 1 の結果より，管理知識をモジュール化した提案システムでは知識ベース全体の知識構成を把握していなくても，管理知識を含むモジュールの単位での操

作により，システム構成の変化の原因となった ICT サービスに焦点を当てた，部分的な知識ベースの更新を容易に行えることを確認した．さらに，知識ベース全体の知識構成を熟知している管理者でなければ知識ベースの更新が難しいという，従来の KNMS に関する課題を確認した．また，診断実行時の AIR/エージェント間メッセージ数の比較より，提案システムでは AIR/エージェント間メッセージの制御，すなわち，管理知識の整合性管理にかかる管理者負担が軽減されたことを確認した．

実験 2 では，知識ベース更新のシミュレーションを行った．シミュレーションの結果より，提案システムでは ICT サービスの数が増えた場合でも，既存システムと比較して知識ベースの更新にかかる管理者負担が小さく抑えられることを確認した．

以上より，提案（S1）モジュール化に基づくサービス指向型の知識管理法 により，課題（P1）知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難 が解決可能であることが示された．ゆえに，本提案に基づく KNMS は変動するネットワークシステムにおいても，自律的な障害管理の実現に貢献可能であると言える．

3.5　まとめ

本章では，変動するネットワークシステムに適用可能な KNMS の実現に向け，特に自律的な障害管理に焦点を当て，2 章にて述べた課題（P1）の解決を目指し，（S1）モジュール化に基づくサービス指向型の知識管理法 を提案した．本提案では管理知識をサービス毎にモジュール化することで，システム構成の変化の原因となったサービスに焦点を当てた知識ベースの更新を可能とした．これにより，知識ベース全体の知識構成を把握していない管理者による知識ベースの更新を可能とした．また，試作システムを設計・実装し，実験を行うことで，本提案を導入し

たKNMSが，従来のKNMSと比較して容易に知識ベースを更新可能であることを確認した．以上より，提案（S1）により課題（P1）が解決可能であることを示し，変動するネットワークシステムにおいても自律的な障害管理が可能なKNMSの実現可能性を示した．

第4章　知識型ネットワーク管理システムによる自律的な性能管理

第 4 章では，知識型ネットワーク管理システムを用いた性能管理タスクの自律化に焦点を当て，ネットワークシステムの全体的な運用状態を把握し，この結果を運用管理タスクに反映させる新しい手法を提案し，システム構成の変化に強い汎用的な管理知識をどのように獲得・記述すべきかについて述べる．

4.1　概要

近年の仮想化技術の成熟を背景としたクラウド・コンピューティング技術の進展により，ハードウェア構成に束縛されない柔軟な ICT サービスの構築が可能となった．3 章で述べたとおり，仮想化技術に基づいて構築されたサービスの大きな特徴の一つとして，柔軟性が挙げられる．特に，サービスの規模を柔軟に拡張・縮小させることで自律的に利用者需要の量的な変動に対応する手法やサービスは，Elastic Resource Management や Elastic Services などと呼ばれている [47, 48, 49]．このような仮想化環境の特徴である柔軟性を活かすためには，不規則に到来するサービス処理要求に伴って刻々と変化するシステムの運用状態を的確に把握して，システムの制御を行う必要がある．現在，そのための様々な研究が，理論的モデルの分析から実システムの動作特性の観測・制御に至る広範な領域で進められている．例えば，稼働中のシステムやサービスの動作状況を把握する際に，スループットやレスポンスタイムに代表される QoS が広く利用されている．サーバシステムが提供するサービスの QoS の低落が検出された場合には，システムが過負荷状態

にあると推察して，これを解消する制御動作を発動したりすることができる．しかし，こうした QoS の変化が検出されるのは，過負荷や資源枯渇，あるいは，障害など，サーバシステムにとって不都合な状態が既に発生した状態となった後となることから，こうした不都合な状態に陥りつつあることを事前に察知したり，これを回避したりすることには難しい．従って，安定的なサービス運用を目指すためには，システムの運用状態の観測に基づいて，システムが不都合な状況に陥りつつあることを検出するための新たな手段が必要となる．

2 章にて述べたとおり，自律的な性能管理の実現に向けた KNMS に関する取り組みには，(P2) 動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 という課題により，その適用範囲を，構成の変化しない固定的なネットワークシステムに限定されてきた．本章では，この課題に対して，(S2) 巨視的な観点に基づくネットワークシステムの内部状態の観測法 を提案することで，変動するネットワークシステムにおいて頻繁に変化する動作特性によらず，その運用状態を捉え，システムが不都合な状況に陥りつつあることを検出する手段を実現する．

4.2　関連研究と課題

仮想化環境の持つ柔軟性に基づいてネットワークシステムの制御を行う際には，変動し続ける利用者需要に合わせてネットワークシステムに与える計算資源の量を調整することにより，サービスを停止させること無く安定的に運用することが重要である．この時，利用者の需要の大きさとサービスに与える計算資源の量は，図 4.1 に示すようなトレードオフの関係にある．すなわち，サービスの需要に対して計算資源の量が不足している状況では，サービスは過負荷状態に陥り，一方，計算資源が過剰となる状況では，サービスに割り当てた計算資源に無駄が発生し，費用や消費電力の観点から望ましくない．従って，このような状況を避けるため

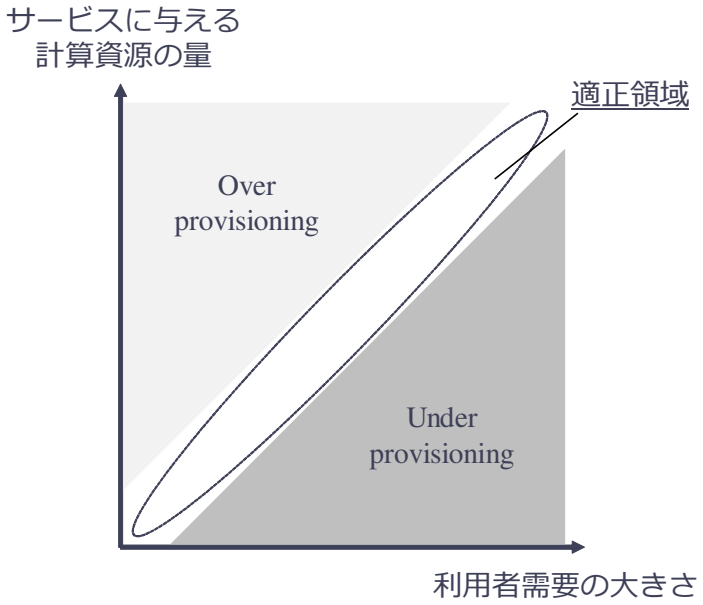


図 4.1: 利用者需要の大きさに合わせた計算資源の管理

には，サービスの運用状態を把握して適切なタイミングでシステムに与える計算資源の量を調整しなければならないが，こうしたタイミングを適切に判断することは難しい．これは，ネットワークシステムの動作特性が，サービスの運用状態の変化に伴って不規則に変動するからである．

2 章にて述べたとおり，従来の KNMS に関する取り組みでは効果的な性能管理の実現に向け，サービスの運用状態を捉えるためにシステムの動作特性に関する詳細な事前知識を必要としていた．しかし，このようなネットワークシステムの動作特性，特に高負荷時のシステムのふるまいは，実際にシステムに大きな負荷を与え，過負荷状態を引き起こしてみなければ，知り得ない情報である．ゆえに，本研究で焦点を当てる変動するネットワークシステムにおいては，構成が変化する度にシステムの動作特性を計測し直す必要があるが，それにはサービスの停止を伴うため，実運用上困難である．よって，(P2) 動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 という課題は，変動するネットワークシステムへの適用が可能な KNMS を実現する上で解決する

必要がある．

4.3　巨視的な観点に基づくネットワークシステムの内部状態の観測法

4.3.1　提案の概要

上述したネットワークシステムの性能管理に関する研究開発の現状から分かるように，ネットワークシステムの運用状態を的確に把握するための手法，特に，システムが不都合な状況に陥ることを回避するために有用な手法を確立することが重要である．さらに，サービスに与える計算資源の量が頻繁に変動する場合，これに伴って発生するネットワークシステム自体の動作特性の変動による影響を考慮する必要がある．これは，構成変化後のシステムの運用状態を把握するために動作特性の再取得が必要となる場合，多大なオーバヘッドが発生するからである．従って，ネットワークシステムの状態観測のための新たな手法を確立するためには，変動する動作特性に強く依存しない手法を工夫する必要がある．

そこで，本研究では，ネットワークシステムの運用状態の急激な変化を事前に捉えるための新しい観測手法について検討する．具体的には，システムの動作特性の観測に関する先行研究 [50] で得られた知見をもとに，ネットワークシステムの巨視的なふるまいを表す「活動度」に含まれるゆらぎに着目し，ネットワークシステムの動作モデルを用いたゆらぎ特性の理論解析に基づいて,「活動度のゆらぎの分散」を新しい観測指標として提案する．サービスへの利用者要求の増大やシステム機能障害による処理能力の不足など，何らかの原因によってネットワークシステムへの処理負荷が増大し，システムの処理能力が低落しつつあるとき，ネットワークシステムの動作状態の観測のみでは，この状況を察知することはできない．その理由は，処理能力の低下が，システムが過負荷状態に移行した後にしか観測できない計測量だからである．しかし，本研究で提案する新たな指標は，シス

テムが完全に不都合な状態に陥る前に，すなわち，ネットワークシステムの処理能力の低落が顕在化するのに先立って，指標の観測値が増大するという性質をもつ．さらに，本指標は特定の動作条件やシステム環境に依存しない，汎用的かつ簡便な方法によって観測できる利点もある．そこで，本指標の急激な増大が観測されたときに種々の対策操作を施すことで，システムが過負荷状態に移行する前にその発生を回避することが可能となり，サービスの安定的な運用が実現される．

4.3.2　ネットワークシステムのモデル化

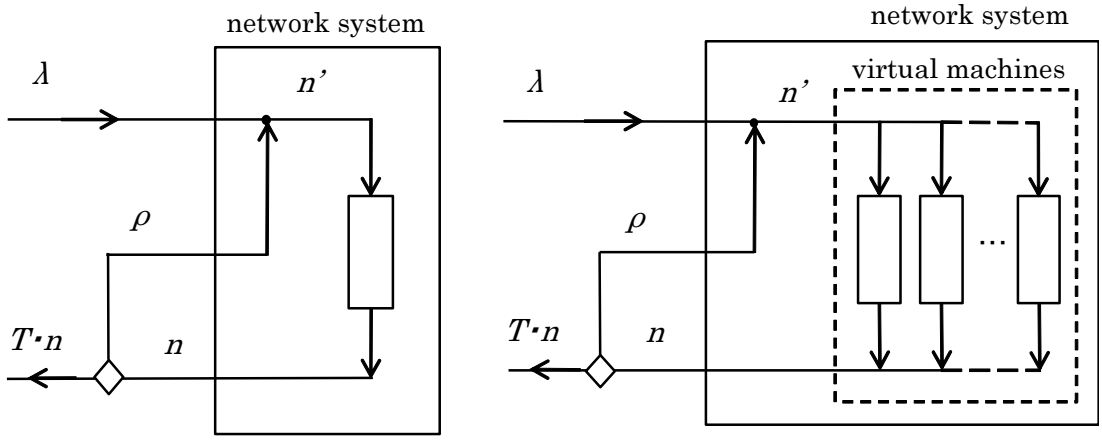
ネットワークシステムの動作の分析について，これまで多くの取り組みが行われてきた．これらの取り組みでは，ネットワークシステムのモデルとして，一般的に待ち行列モデルが用いられている [51, 52]．そこで，本研究ではネットワークシステムの巨視的な動作を分析するため，従来の待ち行列モデルに基づき，単一サーバから構成されるネットワークシステムのモデルを図 4.2(a) に示すとおりに定義する．このモデルは，ネットワークシステムの巨視的な動作，すなわち，ネットワークシステムへの入出力に着目したとき，図 4.2(b) に示すとおり，多数のハードウェア・ソフトウェアにより構成され，絶えず構成の変化するネットワークシステムにも適用することができる．

ある時刻 t に到来するデータ量を λ，ネットワークシステムで処理される量を n，再処理される割合と出力される割合をそれぞれ ρ および T とし，そして時刻 $t+1$ にネットワークシステムで処理される量を n' とすると，図 4.2 に示すネットワークシステムのふるまいは次のようになる．

$$\begin{aligned} n' &= \lambda + \rho \cdot n \\ n &= \rho \cdot n + T \cdot n \end{aligned} \tag{4.1}$$

このとき，時刻 t に処理されるデータ量の割合 W は，

$$W\left(n \to n+1\right) = \lambda + \rho \cdot n \tag{4.2}$$



(a) 単一窓口待ち行列モデル　　(b) 変動するネットワークシステムのモデル

λ： arrived data
n： processed data at time t
n': data to be processed at time t+1
ρ： reprocessing ratio of data
T： throughput
$T \cdot n$： amount of processed data

図 4.2: ネットワークシステムのモデル

$$W\left(n \to n-1\right) = \rho \cdot n + T \cdot n \tag{4.3}$$

と定義することができる．

ここで，時刻 t においてネットワークシステムが状態 n をとる確率分布を $P(n,t)$ とすると，次の状態遷移方程式が成り立つ．

$$\begin{aligned} \frac{[P(n,t+1)-P(n,t)]}{t} &\cong \frac{\partial}{\partial t}P(n,t) \\ = -[(\lambda+\rho\cdot n)+(\rho\cdot n+T\cdot n)]\cdot P(n,t) & \\ +[\lambda+\rho\cdot(n-1)]\cdot P(n-1,t) & \\ +[(n+1)\cdot(\rho+T)]\cdot P(n+1,t) & \end{aligned} \tag{4.4}$$

（4.4）式に関して，第 1 項は微少時間 t において，状態 n から状態 $n+1$ および状態 $n-1$ へ遷移する際の影響を表しており，その確率は増加の割合 $(\lambda+\rho\cdot n)$，および，減少の割合 $(\rho\cdot n+T\cdot n)$ に比例する．そして，第 2 項と第 3 項は，それぞれ状態 $n+1$ と状態 $n-1$ から状態 n へ遷移する際の影響を表している．

ネットワークシステムのデータ処理量について，その最大許容量を N，ネットワークシステムの活動度を x とし，$x=n/N$，$\mu=\lambda/N$，$\varepsilon=1/N(\ll 1)$ とすれば，(4.4）式は次のようになる．

$$\begin{aligned} \varepsilon\cdot\frac{\partial}{\partial t}P(x,t) = -[\mu+\rho\cdot x+(\rho+T)]\cdot P(x,t) & \\ +[\mu+\rho\cdot(x-\varepsilon)]\cdot P(x-\varepsilon,t) & \\ +[(x+\varepsilon)\cdot(\rho+T)]\cdot P(x+\varepsilon,t) & \end{aligned} \tag{4.5}$$

活動度 x の定常状態を x_0 とし，活動度 x のゆらぎを

$$\xi = \frac{1}{\sqrt{\varepsilon}}\cdot(x-x_0) \tag{4.6}$$

と表すと，(4.5）式および [53] より，活動度 x のゆらぎ ξ に関する Fokker-Plank 方程式が以下の手続きより得られる．

まず，(4.5）式を ε のまわりでテイラー展開すると，

$$\begin{aligned}\frac{\partial}{\partial t}P(x,t) = \sum_{n=1}^{\infty}\frac{1}{n!}\cdot(-1)^n\cdot\varepsilon^{n-1}\cdot\frac{\partial^n}{\partial x^n}[(\mu+\rho\cdot x)\cdot P(x,t)] \\ +\sum_{n=1}^{\infty}\frac{1}{n!}\cdot\varepsilon^{n-1}\cdot\frac{\partial^n}{\partial x^n}[(\rho+T)\cdot x\cdot P(x,t)]\end{aligned} \tag{4.7}$$

ここで，連鎖律より，

$$\frac{\partial}{\partial\tau} = \frac{\partial\xi}{\partial\tau}\cdot\frac{\partial}{\partial\xi}+\frac{\partial t}{\partial\tau}\cdot\frac{\partial}{\partial t} = \left[-\frac{1}{\sqrt{\varepsilon}}\cdot\frac{\partial x_0}{\partial\tau}\right]\cdot\frac{\partial}{\partial\xi}+\frac{\partial}{\partial\tau} \tag{4.8}$$

$$\frac{\partial}{\partial x} = \frac{\partial\xi}{\partial x}\cdot\frac{\partial}{\partial\xi}+\frac{\partial t}{\partial x}\cdot\frac{\partial}{\partial t} = \frac{1}{\sqrt{\varepsilon}}\cdot\frac{\partial}{\partial\xi} \tag{4.9}$$

これより（4.7）式は以下のようになる．

$$\begin{aligned}&-\frac{1}{\sqrt{\varepsilon}}\cdot\frac{\partial x_0}{\partial t}\cdot\frac{\partial}{\partial\xi}P(\xi,t)+\frac{\partial}{\partial t}P(\xi,t) \\ &=\sum_{n=1}^{\infty}\frac{1}{n!}\cdot(-1)^n\cdot\varepsilon^{\frac{n}{2}-1}\cdot\frac{\partial^n}{\partial\xi^n}\left[\sum_{m=0}^{\infty}\frac{1}{m!}\cdot(\sqrt{\varepsilon}\cdot\xi)^m\cdot\frac{\partial^m}{\partial x_0^m}[(\mu+\rho\cdot x_0)\cdot P(\xi,t)]\right] \\ &+\sum_{n=1}^{\infty}\frac{1}{n!}\cdot\varepsilon^{\frac{n}{2}-1}\cdot\frac{\partial^n}{\partial\xi^n}\left[\sum_{m=0}^{\infty}\frac{1}{m!}\cdot(\sqrt{\varepsilon}\cdot\xi)^m\cdot\frac{\partial^m}{\partial x_0^m}[(\rho+T)\cdot x_0\cdot P(\xi,t)]\right]\end{aligned} \tag{4.10}$$

（4.10）式について，ε^0 の項は，$n=2, m=0$ および $n=1, m=1$ の条件のもと導かれる．

$$\begin{aligned}&\frac{\partial}{\partial t}P(\xi,t) = \frac{1}{2}\cdot\frac{\partial^2}{\partial\xi^2}[(\mu+\rho\cdot x_0)\cdot P(\xi,t)]+\frac{1}{2}\cdot\frac{\partial^2}{\partial\xi^2}[(\rho+T)\cdot x_0\cdot P(\xi,t)] \\ &-\frac{\partial}{\partial\xi}\left[\xi\cdot\frac{\partial}{\partial x_0}[(\mu+\rho\cdot x_0)\cdot P(\xi,t)]\right]+\frac{\partial}{\partial\xi}\left[\xi\cdot\frac{\partial}{\partial x_0}[(\rho+T)\cdot x_0\cdot P(\xi,t)]\right] \\ &=\frac{1}{2}\cdot[(\mu+\rho\cdot x_0)+(\rho+T)\cdot x_0]\cdot\frac{\partial^2}{\partial\xi^2}P(\xi,t)+\frac{\partial}{\partial x_0}(T\cdot x_0)\cdot\frac{\partial}{\partial\xi}[\xi\cdot P(\xi,t)]\end{aligned} \tag{4.11}$$

以上より，活動度 x のゆらぎ ξ に関する次の Fokker-Plank 方程式が得られる．

$$\frac{\partial}{\partial t}P(\xi,t) = \frac{1}{2}\cdot[(\mu+\rho\cdot x_0)+(\rho+T)\cdot x_0]\cdot\frac{\partial^2}{\partial\xi^2}P(\xi,t)+\frac{\partial}{\partial x_0}(T\cdot x_0)\cdot\frac{\partial}{\partial\xi}[\xi\cdot P(\xi,t)] \tag{4.12}$$

さらに，(4.10）式について，$\varepsilon^{-\frac{1}{2}}$ の項は，$n=1, m=0$ のもと，

$$\frac{\partial x_0}{\partial t} = (\mu + \rho \cdot x_0) - (\rho + T) \cdot x_0 \tag{4.13}$$

これより，x_0 の時間発展として，

$$\frac{\partial x_0}{\partial t} = \mu - T \cdot x_0 \tag{4.14}$$

が得られる．

4.3.3　ゆらぎの分析に基づくネットワークシステムの状態観測

活動度のゆらぎの分散 σ_ξ^2 は

$$\sigma_\xi^2 = \langle \xi^2 \rangle - \langle \xi \rangle^2$$

$$\frac{\partial}{\partial t}\sigma_\xi^2 = \frac{\partial}{\partial t}\langle \xi^2 \rangle - 2\langle \xi \rangle \cdot \frac{\partial}{\partial t}\langle \xi \rangle \tag{4.15}$$

で与えられる．

（4.12）式に示す Fokker-Plank 方程式より，(4.15）式の第 1 項および第 2 項は以下のようになる．

$$\frac{\partial}{\partial t}\langle \xi^2 \rangle = \int \xi^2 \cdot \frac{\partial}{\partial t} P(\xi, t) d\xi$$

$$= \frac{1}{2} \cdot [\mu + (2\rho + T) \cdot x_0] \cdot \int \xi^2 \cdot \frac{\partial^2}{\partial \xi^2} P(\xi, t) d\xi + \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \int \xi^2 \cdot \frac{\partial}{\partial \xi}[\xi \cdot P(\xi, t)] d\xi$$

$$= [\mu + (2\rho + T) \cdot x_0] - 2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \langle \xi^2 \rangle \tag{4.16}$$

$$\frac{\partial}{\partial t}\langle \xi \rangle = \int \xi \cdot \frac{\partial}{\partial t} P(\xi, t) d\xi$$

$$= \frac{1}{2} \cdot [\mu + (2\rho + T) \cdot x_0] \cdot \int \xi \cdot \frac{\partial^2}{\partial \xi^2} P(\xi, t) d\xi + \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \int \xi \cdot \frac{\partial}{\partial \xi}[\xi \cdot P(\xi, t)] d\xi$$

$$= -\langle \xi \rangle \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \tag{4.17}$$

よって，

$$\frac{\partial}{\partial t}\sigma_\xi^2 = \frac{\partial}{\partial t}\langle\xi^2\rangle - 2\langle\xi\rangle \cdot \frac{\partial}{\partial t}\langle\xi\rangle$$
$$= [\mu + (2\rho + T) \cdot x_0] - 2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \langle\xi^2\rangle - 2\langle\xi\rangle \cdot \left(-\langle\xi\rangle \cdot \frac{\partial}{\partial x_0}(T \cdot x_0)\right)$$
$$= [\mu + (2\rho + T) \cdot x_0] - 2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot (\langle\xi^2\rangle - \langle\xi\rangle^2)$$
$$= [\mu + (2\rho + T) \cdot x_0] - 2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \sigma_\xi^2 \tag{4.18}$$

以上より，

$$\frac{\partial}{\partial t}\sigma_\xi^2 = [\mu + (2\rho + T) \cdot x_0] - 2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0) \cdot \sigma_\xi^2 \tag{4.19}$$

となる．

また，定常状態に着目すると次の条件が満たされる．

$$\frac{\partial x_0}{\partial t} = \frac{\partial \sigma_\xi^2}{\partial t} = 0$$

この条件のもと，(4.14）式と（4.19）式より，定常状態の最確値 x_0 および活動度のゆらぎの分散 σ_ξ^2 は，

$$T \cdot x_0 - \mu = 0 \tag{4.20}$$

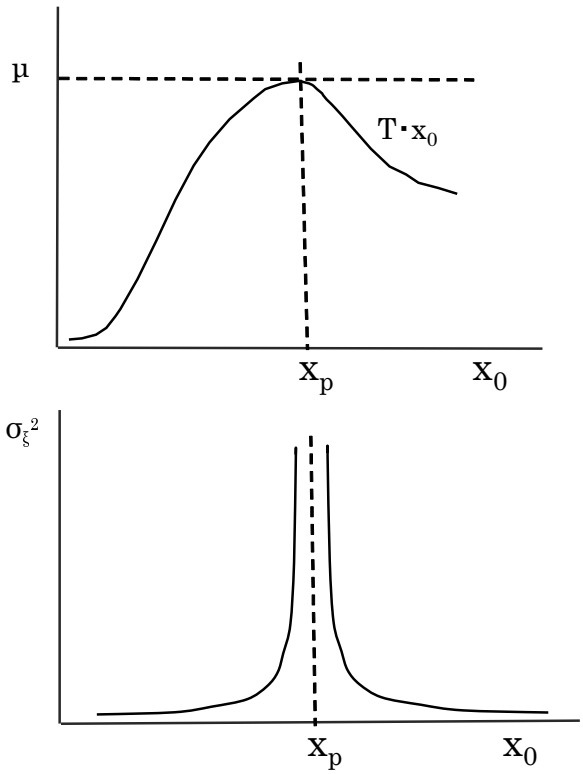
$$\sigma_\xi^2 = \frac{\mu + (2\rho + T) \cdot x_0}{2 \cdot \frac{\partial}{\partial x_0}(T \cdot x_0)} \tag{4.21}$$

で与えられる．

ネットワークシステムに到来するデータ処理要求の量の変動，すなわち，ネットワークシステムにかかる負荷の変動により，ネットワークシステムのスループット特性 T は度々変化する．第 2.1.3 項にて述べたとおり，負荷の増大に対処するためには，サービスに与える計算資源の量を増やすなどの処置を施すことにより，システムが許容できる負荷の上限を調整する必要がある．もし，負荷の大きさが最大許容量を上回った場合，システムのスループットの大きさ，すなわち，QoS の値は破局的に低下する．さらに，QoS の破局的な低下が発生した後の状態では，た

とえ負荷の大きさがシステムの最大許容量付近まで減少したとしても QoS は回復せず，最大許容量をさらに下回り，システムの処理能力に大きな余裕を生じるまで QoS が回復することは無い．ゆえに，ネットワーク管理者は安定的にサービスを運用するために，ネットワークシステムにかかる負荷の大きさに関して，いつ最大許容量に到達するか，そして QoS の低下が始まるかを把握しておかなければならない．しかし，このような動作特性は，ネットワークシステムに含まれるハードウェア構成やソフトウェア構成など，様々な要因で変化するため，あらかじめ把握しておくことは困難である．この問題を解決するため，本研究では，巨視的な観点から眺めたときのネットワークシステムのふるまい，すなわち，ネットワークシステムの活動度について，そのゆらぎ成分の分散を観測・分析することで，スループットの破局的な低落を予測するための手法を提案する．

図 4.3 に示すとおり，ネットワークシステムのデータ処理量が極値をとり，最大許容値 μ と等しくなるとき，ネットワークシステムの活動度のゆらぎの分散 σ_ξ^2 が発散する．さらに，(4.20) 式および (4.21) 式より明らかなように，$\frac{\partial}{\partial x_0}(T \cdot x_0) \to 0$ となるとき，ゆらぎの分散 σ_ξ^2 は分母が 0 に近づき，$\sigma_\xi^2 \to \infty$ となる．よって，ネットワークシステムの活動度のゆらぎの分散 σ_ξ^2 の異常増大を観測することで，負荷の大きさが最大許容量に近いかどうか，QoS の急激な低下が発生する直前かどうかを判定することが可能になる．以上のとおり，運用中のネットワークシステムの活動度のゆらぎの分散を，ネットワークシステムの運用状態の急激な変化を事前に捉えるための指標として提案する．



x_0： steady state of activity factor
σ_ξ^2： variance of fluctuation of x_0
T： throughput
$T \cdot x_0$： amount of processed data
μ： maximum value of $T \cdot x_0$

If $x_0 = x_p$ *then*
$T \cdot x_0$ *is equal to* μ *and*
$\partial(T \cdot x_0) / \partial x_0 = 0$.
Hence,
$\sigma_\xi^2 \to \infty$ when $\partial(T \cdot x_0) / \partial x_0 \to 0$.

図 4.3: ネットワークシステムの活動度のゆらぎに関する特性

4.4　評価実験

4.4.1　実験概要

前節にて提案した新しい指標「ネットワークシステムの活動度のゆらぎの分散」は，ネットワークシステムの動作状況の把握にとって有用な特性，すなわち,「稼働中のネットワークシステムが不都合な状態に陥りつつあるとき，システムが完全に不都合な状態に遷移するのに先行して，本指標が急激に増大する」という特性を持つ．そこで，実機のネットワークシステムを用いた実験によってその特性を評価し，ネットワークシステムの観測指標としての有効性を検証する．

図 4.4 に実験システムの構成を示す．本実験では，仮想化プラットフォームである VMware vSphere [21] を使用してサービスを構築する．また，ネットワークシステム上で稼働するサービスとして，データベースと連動して動的に Web ページを生成するような Web アプリケーションを使用する．ネットワークシステムは初期状態において，1 台の仮想マシン（VM）で構成されており，必要に応じて計算資源の追加（スケーリング）を行う．スケーリング手法はロードバランサを介して複製された VM を並列に接続する，horizontal scaling [54] を用いる．この Web サービスへの HTTP リクエストはロードバランサを介して各 VM に振り分けられる．リクエストの振り分けは，各 VM の負荷が均等になるよう，VM のコア数に応じた重み付きラウンドロビンに基づいて行う．

前節にて定式化した手法に基づいて，ネットワークシステムの活動度のゆらぎの分散の時間的な推移を観測するため，動作状態の観測ツールとして Apache Bench [55] を使用する．ネットワークシステムに対する処理負荷は，ネットワークシステムに到来する単位時間あたりの HTTP リクエスト数によって調整され，これらのリクエストを処理するネットワークシステムの活動度は，単位時間あたりに処理されるリクエスト量を表す transfer rate により観測される．これらの観測値は Apache Bench

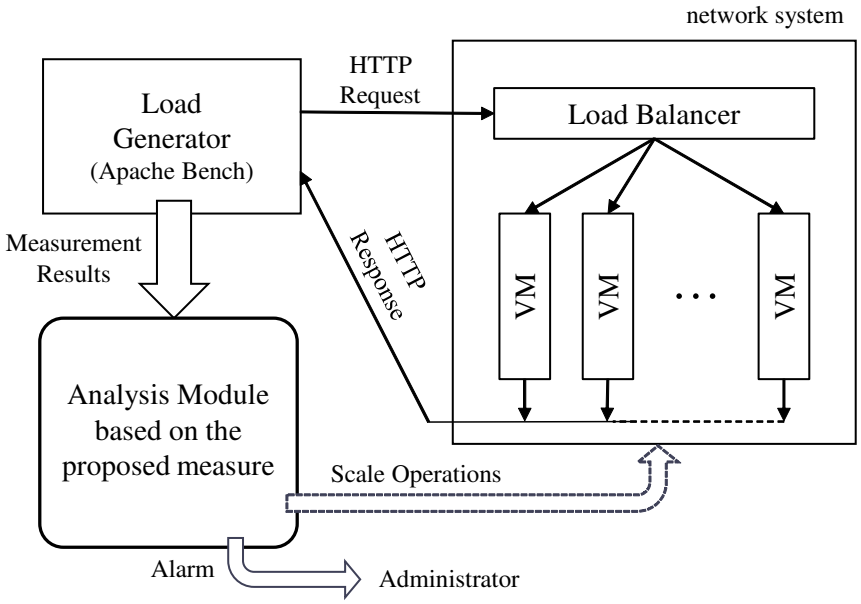


図 4.4: 実験システムの構成

で取得可能である．そして，以下の手続きに基づいて，時刻 t における transfer rate のゆらぎの分散 $var(t)$ が，計測間隔 ΔT 秒毎に算出される．

1. 時刻 t における transfer rate 計測値 v_t のゆらぎ $fluc(t)$ を算出し，これに平滑化を施した $fluc'(t)$ を得る．

$$fluc(t) = v_t - \frac{1}{L}\sum_{i=0}^{L-1} v_{t-i} \tag{4.22}$$

$$fluc'(t) = \frac{1}{M}\sum_{i=0}^{M-1} fluc(t-i) \tag{4.23}$$

2. 次式により，時刻 t における transfer rate のゆらぎの分散 $var(t)$ を算出する．

$$var(t) = \frac{1}{N}\sum_{i=0}^{N-1} \left(fluc'(t-i)\right)^2 - \left(\frac{1}{N}\sum_{i-0}^{N-1} fluc'(t-i)\right)^2 \tag{4.24}$$

本実験では計測間隔を $\Delta T = 1$ とし，平滑化のための移動平均の点数として，$L = 10$，$M = 10$，分散計算の対象とする点数として，$N = 10$ と設定した．

まず，前節で述べた作業仮説の検証のため，以下の 3 通りの実験を行う．

1. 実験 3-1：後に示す実験 3-2 および実験 3-3 との比較を行うため，初めにネットワークシステムの正常状態における transfer rate を計測し，そのゆらぎの分散を算出する．正常状態におけるネットワークシステムの挙動を観測するため，負荷の大きさをネットワークシステムの性能限界を超えない範囲に設定し，transfer rate の計測を行う．図 4.3 に示すとおり，ネットワークシステムにかかる負荷が小さく，すなわち $x_0 < x_p$ となる場合，算出される transfer rate のゆらぎの分散は 0 に近い値を示すと予想される．

2. 実験 3-2：ネットワークシステムにかける負荷の大きさを時間と共に増大させることで，過負荷状態を発生させる．この際の transfer rate を計測し，そのゆらぎの分散の変化を確認する．図 4.3 に示すとおり，ネットワークシステムが性能限界に到達する付近，すなわち $x_0 = x_p$ となる付近では，transfer rate のゆらぎの分散が異常増大すると予想される．さらに，性能限界を超過し，$x_0 > x_p$ となる場合，算出される transfer rate のゆらぎの分散は $x_0 < x_p$ の場合と同様に 0 に近い値を示すと考えられる．

3. 実験 3-3：実験 3-2 とは逆に，過負荷状態から正常状態に復帰する場合について，transfer rate の計測およびゆらぎの算出を行う．図 4.3 に示すとおり，ネットワークシステムが過負荷状態や正常状態，すなわち $x_0 > x_p$ あるいは $x_0 < x_p$ となる場合，transfer rate のゆらぎの分散は 0 に近い値を示すと予想される．また，過負荷状態から正常状態に復帰する場合であっても，$x_0 = x_p$ となる付近では，transfer rate のゆらぎの分散が異常増大すると予想される．

次に，上述した実験で検証された観測指標の特性を活用して，ネットワークシステムが過負荷状態に陥りつつある状態を検出する機能を試作する．そして，本機能を埋め込んだ実験用ネットワークシステムを用いた実験を行う．提案指標を用いることにより，ネットワークシステムが過負荷状態に近づきつつあることを

検出し，transfer rate が低落した状態に陥る前に，サービスのスケーリングを実行できることを検証する．その結果，的確なタイミングでスケーリングを施すことにより過負荷状態が回避され，サービスを安定的に維持できることを示す．

さらに，過負荷状態に陥りつつある状態を検出する機能に加え，提案指標を活用し，ネットワークシステムに与えた計算資源の量が過剰である状態を検出する機能を試作する．これらの機能を実験用ネットワークシステムに適用し，過負荷状態だけでなく計算資源の量が過剰な状態という管理者にとって望ましくない双方の状況を，的確なタイミングで計算資源の量を調整しながら回避可能であるか，すなわち，提案指標を活用した自律的な性能管理が可能であるかを検証する．

4.4.2　実験 3：作業仮説の検証

実験 3-1：正常状態におけるネットワークシステムの活動度のゆらぎの分散の観測

本実験では，ネットワークシステムにかかる平均的な負荷の大きさを，ネットワークシステムの性能限界を超えない範囲，ここでは，平均 11 リクエスト毎秒に設定し，transfer rate を計測した．

実験結果を図 4.5，図 4.6，図 4.7 に示す．図 4.5 はネットワークシステムに与えた負荷（リクエスト数）の推移，図 4.6 は観測された transfer rate の計測値，また，図 4.7 は transfer rate のゆらぎの分散である．

図 4.6 の transfer rate の計測値より，本実験の条件下では transfer rate が 0 付近まで低下する現象は確認されず，すなわち，ネットワークシステムが過負荷状態に陥らず安定的に動作したことが確認できる．また，図 4.7 に示すとおり，transfer rate のゆらぎの分散はほぼ 0 付近で推移している．この結果より，負荷の大きさに対してサービスの計算資源の量に十分余裕がある場合には，前節で述べた解析結果と同様に，ネットワークシステムの活動度のゆらぎの分散が増大しないことが確認された．

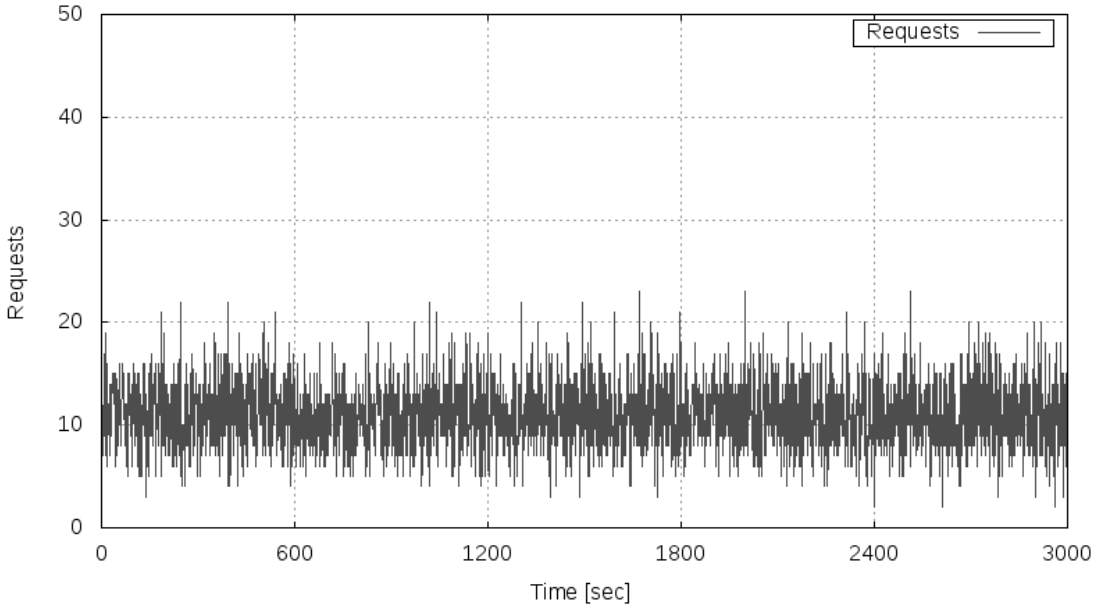


図 4.5: 負荷の推移（実験 3-1）

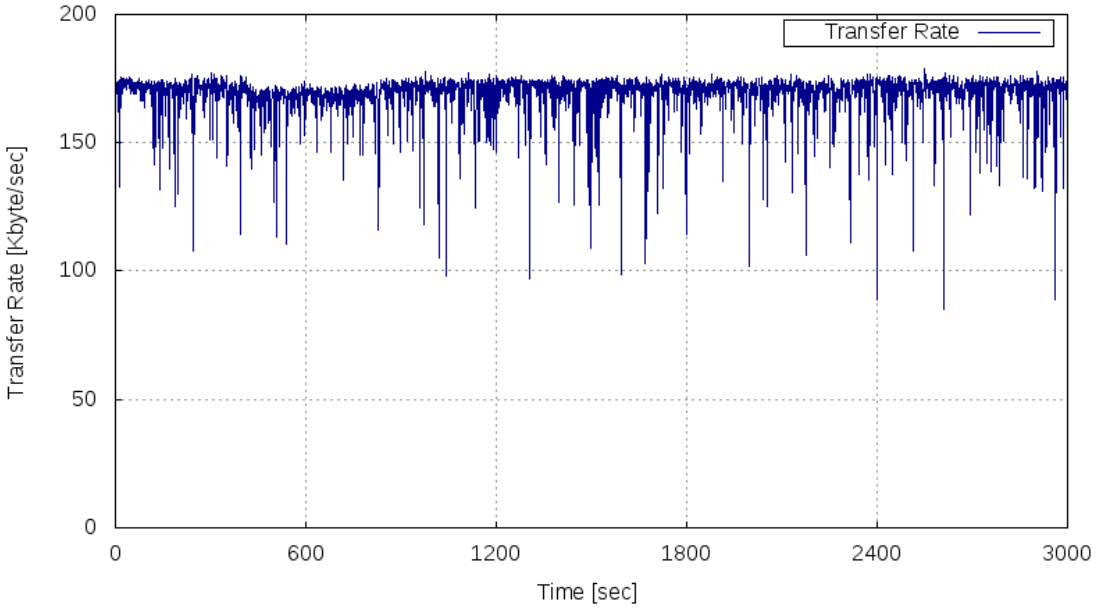


図 4.6: transfer rate の計測値（実験 3-1）

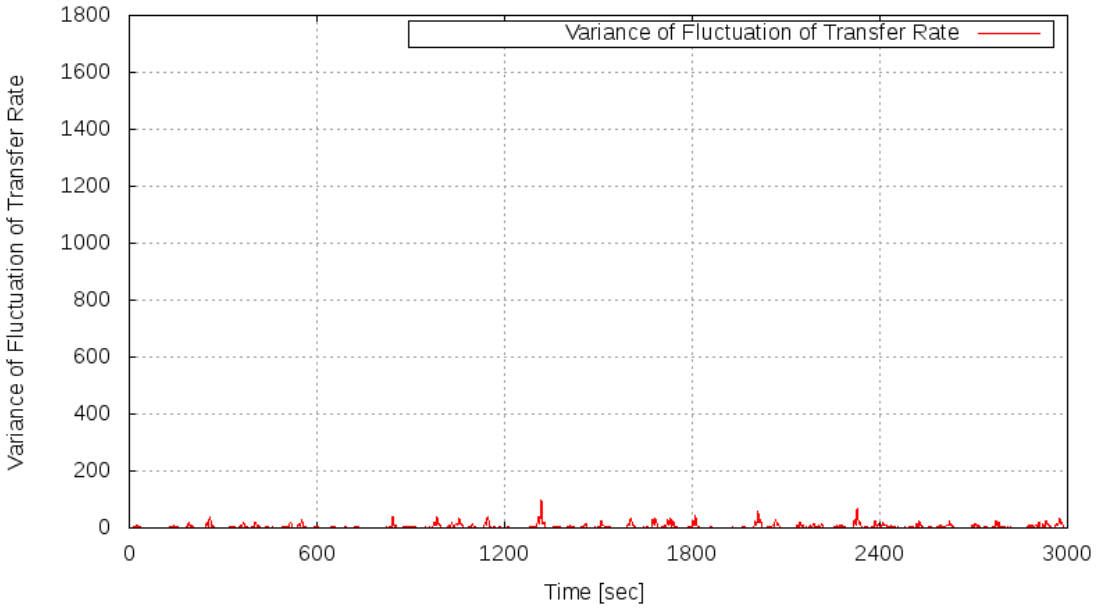


図 4.7: transfer rate のゆらぎの分散（実験 3-1）

実験 3-2：正常状態から過負荷状態への遷移する際のネットワークシステムの活動度のゆらぎの分散の観測

実験 3-2 では，ネットワークシステムにかける負荷の大きさを時間と共に増加させることで過負荷状態を発生させる．前節のゆらぎの特性に関する解析結果より，負荷が増大してネットワークシステムが性能限界に近づいたとき，すなわち，過負荷状態に遷移する直前の状態では，transfer rate のゆらぎの分散が急激に増大し，また，性能限界を超過し完全に過負荷状態に陥った後には，ゆらぎの分散は小さくなると予想される．そこで本実験では，負荷の大きさを，計測開始時点でネットワークシステムが正常に動作すると考えられる平均 10 リクエスト毎秒に設定した．そして，1 秒あたりのリクエスト数は計測開始から 300 秒経過するごとに 1 ずつ増加し，3000 秒時点では平均 20 リクエスト毎秒となる．

実験結果を図 4.8，図 4.9，図 4.10 に示す．図 4.8 はネットワークシステムに与えた負荷（リクエスト数）の推移，図 4.9 は transfer rate の計測値，また，図 4.10 は transfer rate のゆらぎの分散である．

図 4.9 より，0s–1799s の期間では，実験 3-2 の結果と同様に 175Kbyte/sec ほどの transfer rate を維持していることが分かる．また，負荷が増大し，16 リクエスト毎秒となった 1800–2099s の期間において transfer fate が急激に低下する現象が見られ，2100s 以降には transfer rate がほぼ 0 近くにまで低下し，完全に過負荷状態に陥ったことが確認できる．このことから，高負荷時のネットワークシステムのもつ非線形的な動態により，活動度の低下はネットワークシステムが過負荷状態に移行した後にしか観測できないことが分かる．

その一方，図 4.10 に示すゆらぎの分散については，過負荷状態に陥る前の期間，900–1799s の期間において増大傾向にあることが確認できる．さらに，過負荷状態に陥りつつある 1800–2099s の期間では，ゆらぎの分散の増大がより顕著に確認できる．また，過負荷状態に陥った後の，2400s 以降の期間では，再びゆらぎの分散

が 0 付近で推移している．以上より，負荷が増大しネットワークシステムが性能限界に近づくときに transfer rate のゆらぎの分散が急激に増大すること，そして，サーバシステムが過負荷状態に陥った後にゆらぎの分散が収束することが確認された．これは，予想に示したとおりの結果である．

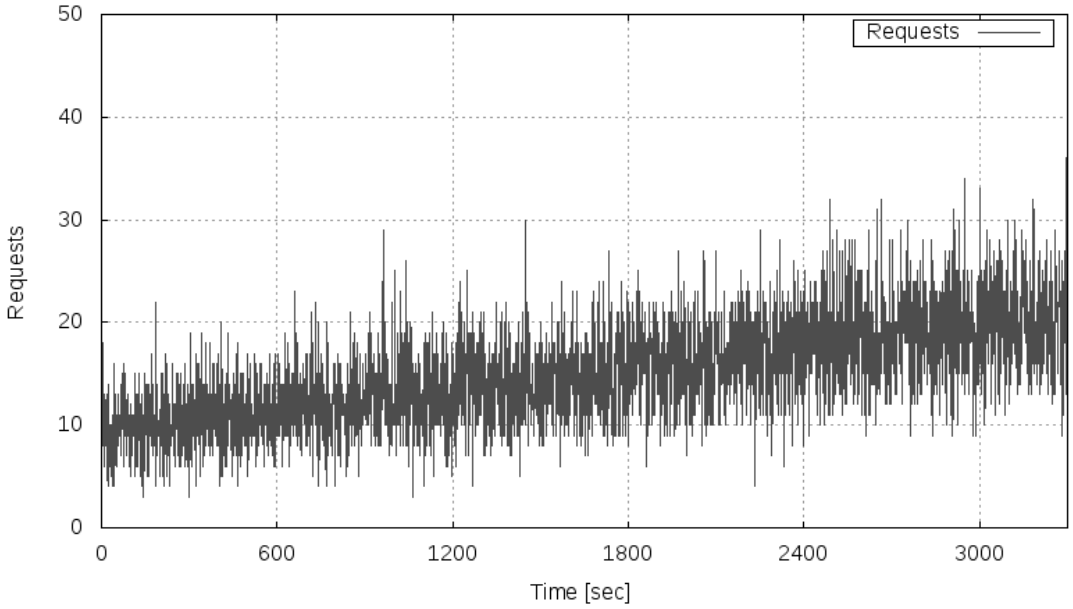


図 4.8: 負荷の推移（実験 3-2）

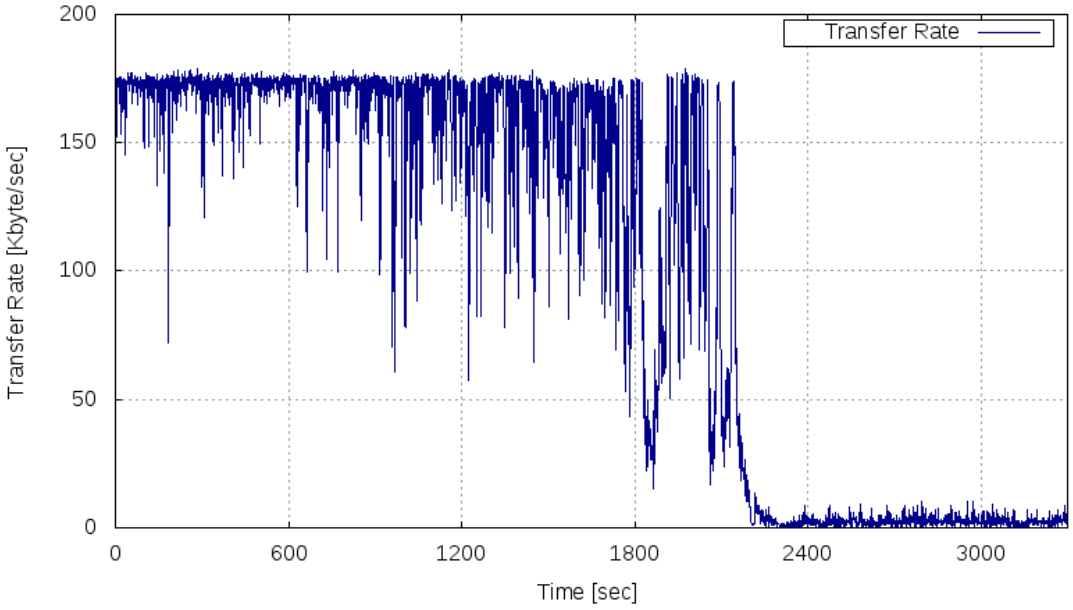


図 4.9: transfer rate の計測値（実験 3-2）

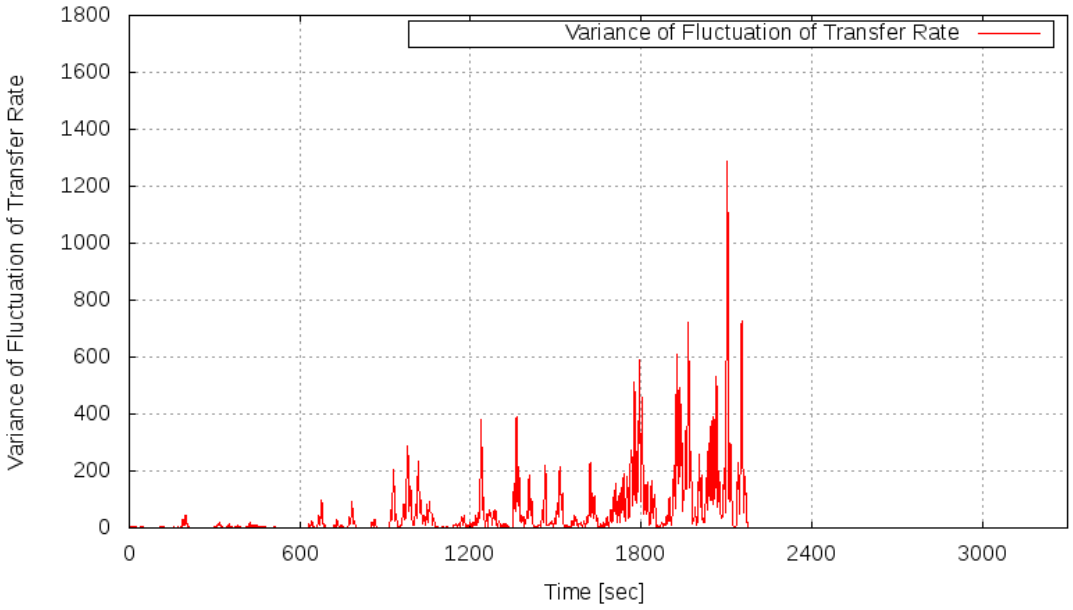


図 4.10: transfer rate のゆらぎの分散（実験 3-2）

実験 3-3：過負荷状態から正常状態への回復する際のネットワークシステムの活動度のゆらぎの分散の観測

実験 3-3 では過負荷状態から正常状態に回復する場合のネットワークシステムの活動度を観測し，そのゆらぎの分散を算出する．すなわち，過負荷状態から負荷が減少し正常状態へ回復する場合の，transfer rate のゆらぎの分散のふるまいについて検証する．前節の解析結果より，実験 3-3 においてもネットワークシステムが性能限界に近づくとき，すなわち正常状態に復帰する直前で，transfer rate のゆらぎの分散が異常増大する現象が生じることが予想される．本実験では，ネットワークシステムにかける負荷の大きさを時間と共に増加させ，一度過負荷状態を発生させた後，再び負荷を徐々に減少させることで正常状態に復帰させる．表 4.1 に，本実験における負荷（リクエスト数）の推移に関する設定を示す．負荷の大きさは計測開始時点で平均 13 リクエスト毎秒とし，以降，300 秒経過する度に，2 ずつ増加させる．そして，過負荷状態に陥った後，900s 時点から，リクエスト数を減少させる．

実験結果を図 4.11，図 4.12，図 4.13 に示す．図 4.11 はネットワークシステムに与えた負荷（リクエスト数）の推移，図 4.12 は transfer rate の計測値，また，図 4.13 は transfer rate のゆらぎの分散である．

表 4.1: ネットワークシステムにかける負荷の設定（実験 3-3）

Time Period [sec]	Number of Request [req/sec]
0–299	13
300–599	15
600–899	17
900–1199	15
1200–1499	13
1500–1799	11
1800–2099	10
2100–2699	9

図 4.12 より，負荷の増大によって，600–899s の期間でネットワークシステムが破局的に過負荷状態に陥ったことが確認できる．その後，負荷の減少に従って 1200–1499s の期間でネットワークシステムが回復し始め，1500s 以降は正常状態に復帰したことが分かる．

一方で，図 4.13 に示す transfer rate のゆらぎの分散については，0–599s の期間で増大傾向にあることが分かる．過負荷状態が発生する予兆として transfer rate のゆらぎの分散が異常増大するこの現象は，実験 3-2 の結果に等しい．過負荷状態に陥っている 600–1199s の期間ではゆらぎの分散が 0 付近で推移しているが，1200–1499s の期間で再びゆらぎの分散が増大していることが確認できる．さらに，1500s 以降は正常状態に回復したことにより，ゆらぎの分散が 0 付近に収束している．以上の結果より，ネットワークシステムが過負荷状態から正常状態に回復する際に，その予兆として transfer rate のゆらぎの分散が増大することが確認できた．よって，本実験においても，ネットワークシステムの活動度のゆらぎの分散について，提案に示すとおりのふるまいが確認できた．

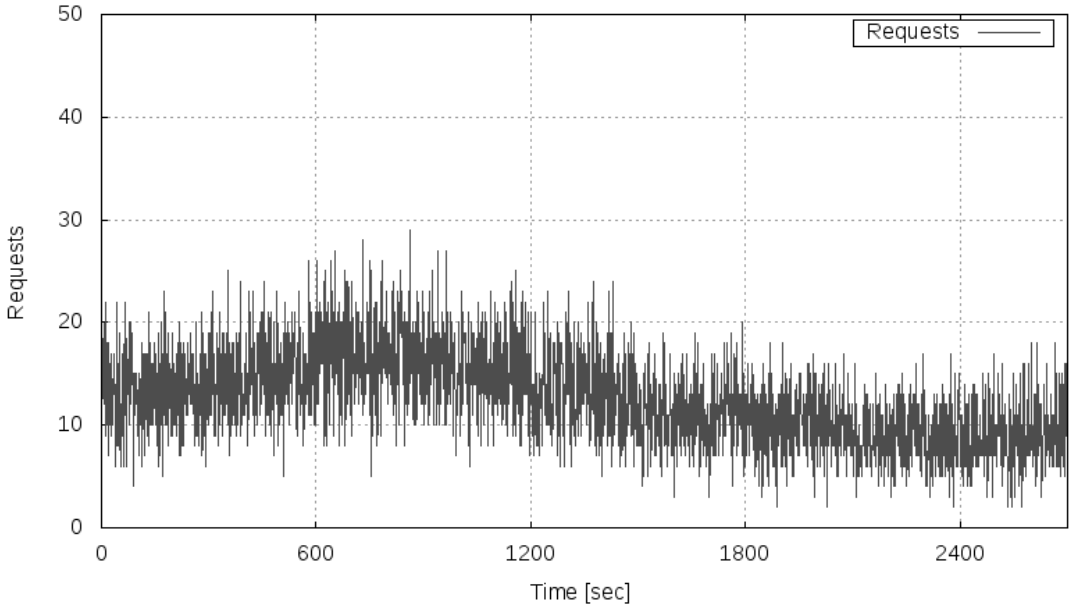


図 4.11: 負荷の推移（実験 3-3）

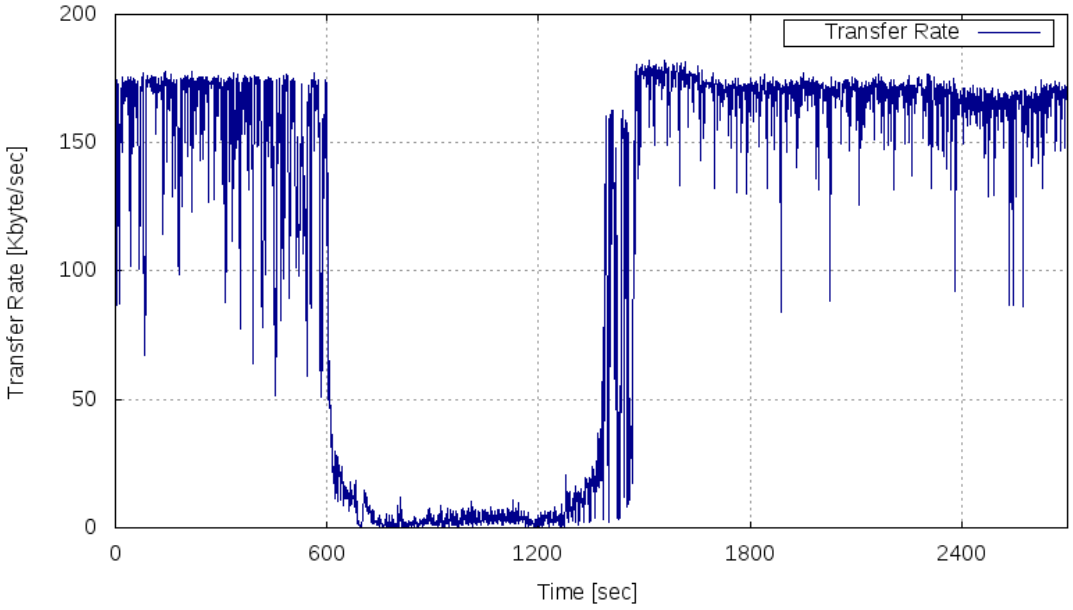


図 4.12: transfer rate の計測値（実験 3-3）

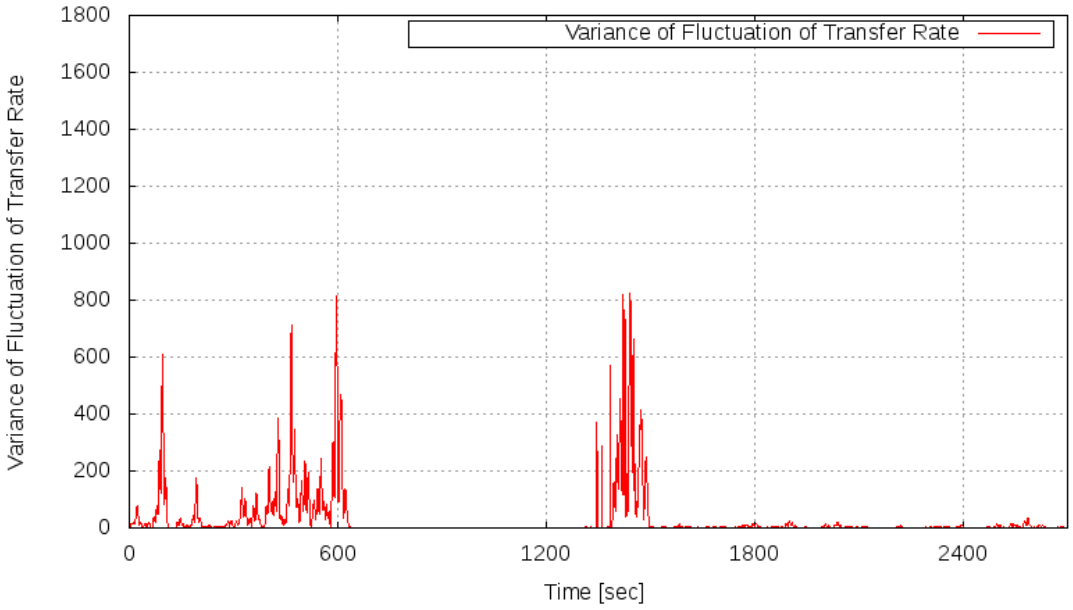


図 4.13: transfer rate のゆらぎの分散（実験 3-3）

以上の実験結果より，提案指標，すなわち「ネットワークシステムの活動度のゆらぎの分散」は前節で述べた解析結果に示すとおりの特性を有することが検証された．実験 3-1 では，ネットワークシステムにかかる負荷が小さく，$x_0 < x_p$ となる場合に，計測される transfer rate のゆらぎの分散が 0 に近い値をとることが確認された．実験 3-2 では，ネットワークシステムにかかる負荷が増大し，処理能力の低下が始まる $x_0 = x_p$ の付近で transfer rate のゆらぎの分散が異常増大することを確認するとともに，性能限界を超えた $x_0 > x_p$ となる場合に，再びゆらぎの分散が 0 付近に収束することを確認した．さらに，実験 3-3 では，ネットワークシステムにかかる負荷の減少により，過負荷状態から正常状態に回復する場合においても，$x_0 = x_p$ となる付近で transfer rate のゆらぎの分散が異常増大することを確認した．これらの結果より，ネットワークシステムの巨視的な動作状態のゆらぎに着目して定式化した提案指標，すなわちネットワークシステムの活動度のゆらぎの分散をサービスの過負荷状態の予兆を検出するための指標として使用できることが示された．

4.4.3　実験 4：適用可能性の検証

実験 4-1：提案指標に基づく過負荷状態の回避

前項までの結果をふまえて，実際のネットワークシステムにおける提案指標の適用可能性を確認するために，提案指標を利用して，ネットワークシステムが過負荷状態に陥りつつある状態を検出する機能を試作した．具体的には，ネットワークシステムの活動度のゆらぎの分散が異常増大している状態が観測された場合に，計算資源の追加を要請するためのアラームを生成する機能を試作し，本機能を実験用システムに組み込んだ．以下に述べる実験では，アラームに基づいてスケーリングを実行し，過負荷状態を回避できるか否かを検証する．本実験で使用する簡易なアラーム生成の手続きを以下に示す．

[時刻 p におけるアラーム生成の手続き]

- 時刻 p において，ネットワークシステムの活動度のゆらぎの分散の算出結果は，配列として $\{var(0), var(1), \ldots, var(p-1), var(p)\}$ で与えられる．
- ゆらぎの分散の算出値について，一つ前のインターバルにて算出された値との差分 d_p は，$d_p \leftarrow var(p) - var(p-1)$ で与えられる．
- $H(H > 0)$ および $C(C > 0)$ は，それぞれゆらぎの分散のピーク値，ピーク値の超過回数に関する既定の閾値である．
- $W(W > 0)$ は，アラーム生成に際して考慮する時間幅を表すウィンドウサイズである．
- S_p は，時刻 $p-1$ において算出されたゆらぎの分散の値がピーク値であり，かつ，既定の閾値を超えているか否かを保持する変数である．
- 各変数は初期状態において，$p = 0$，$d_0 \leftarrow 0$，$S_t \leftarrow 0(-W + 1 \geq t \geq 0)$ であり，以下の手続きによりアラーム生成に関する判定を行う．
- **if** $d_p \geq 0$

  then $S_p \leftarrow 0$

  else{

  if$d_{p-1} \geq 0$ **and** $var(p-1) \geq H$

  then{$S_p \leftarrow 1$

  if $\mathrm{Sum}(S_{p-W+1}, \ldots, S_p) \geq C$ **then** “Generate Alarm”

  }

  else $S_p \leftarrow 0$}

  }

本実験では，$H = 200$，$C = 3$，$W = 180$ とした．すなわち，180s の期間に大きさ 200 を超えるピークが 3 回以上観測されたらアラームを生成するような設定である．アラームが生成された場合には，運用中のサービスを複製した VM を，ロードバランサを介して並列に接続することでサービスのスケーリングを行う．また，負荷の大きさは計測開始時点で平均 10 リクエスト毎秒に設定し，300 秒経過する度に 1 ずつ増加させる．なお，この設定は実験 3-2 と同様である．

実験結果を図 4.14，図 4.15，図 4.16 に示す．図 4.14 はネットワークシステムに与えた負荷（リクエスト数）の推移，図 4.15 は transfer rate の計測値，また，図 4.16 は transfer rate のゆらぎの分散とアラーム生成ログである．

図 4.15 より，本実験では負荷が増大した場合でも transfer rate が 0 付近まで低下するような現象が発生しなかったことが確認できる．1600s および 3600s 付近で transfer rate が上昇していることが分かるが，これはサービスのスケーリングによってネットワークシステムの処理能力が向上したことによるものである．

図 4.16 より，1600s および 3600s 付近でアラームが生成されたことが確認できる．また，アラームを受けた後にサービスのスケーリングを行ったことで，ネットワークシステムが安定的に動作するようになり，ゆらぎの分散が 0 付近まで減少していることが分かる．負荷の大きさについて同一条件で行った実験 3-2 では，1800–2099s の期間で transfer rate が急激に低下する現象が発生していた．その一方で，本実験では transfer rate のゆらぎの分散の観測に基づいて過負荷状態の予兆を検知し，事前に対策操作としてスケーリング処理を施すことができたため，ネットワークシステムが過負荷状態に陥ることを回避できた．このように，提案に基づき試作したアラーム生成機能を用いることで，的確なタイミングでサービスのスケーリングを行い，サービスの過負荷状態を回避可能であることを示した．

さらに，本実験ではスケーリングによりネットワークシステムの動作特性が変化した後にも，アラーム生成機能に再調整を施すことなく同様に過負荷状態の予

兆を検知していることが分かる．ネットワークシステムの詳細な動作特性に基づく事前知識を用いて過負荷状態を予測する場合，例えば，単位時間あたりのクライアント接続数に関する閾値を設定してアラームを生成する場合には，スケーリング後に閾値の再調整が必要となる．このような閾値の再調整は，スケーリング後のネットワークシステムに関する詳細な知識や動作特性を要するため，実運用上困難であるという課題があった．一方，提案指標は変動するネットワークシステムの動作特性によらず，その過負荷状態の予兆を検出することが可能であった．

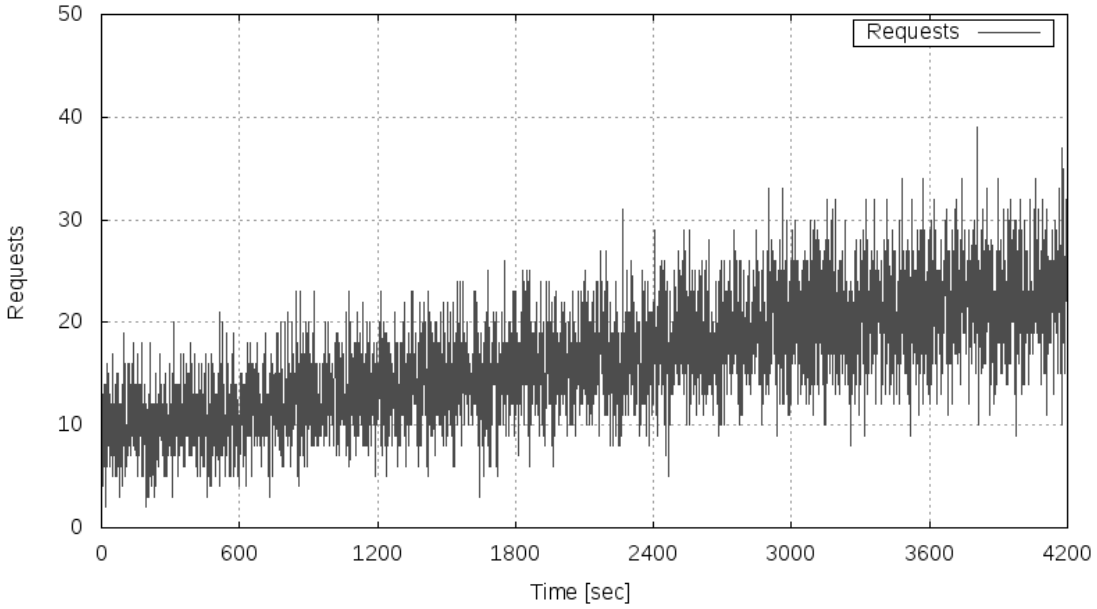


図 4.14: 負荷の推移（実験 4-1）

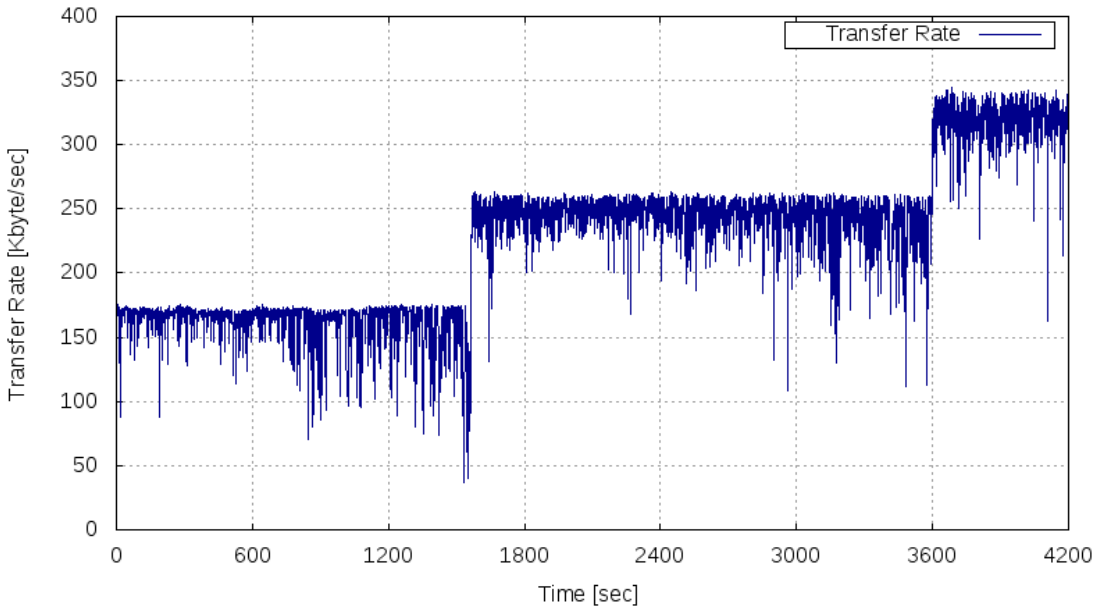


図 4.15: transfer rate の計測値（実験 4-1）

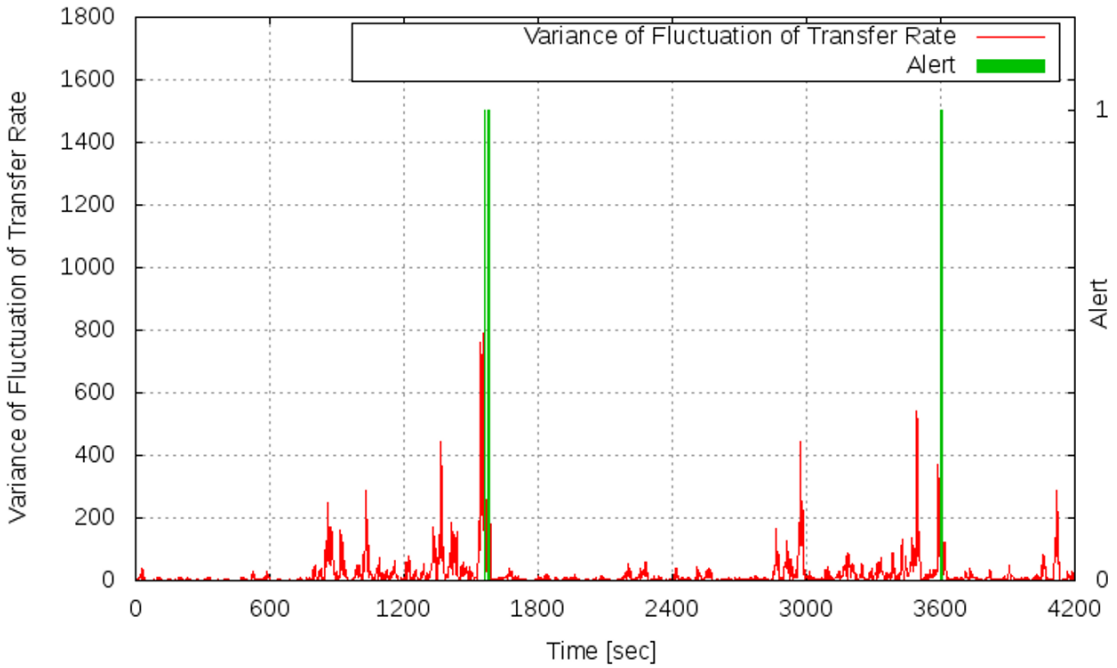


図 4.16: transfer rate のゆらぎの分散とアラーム生成ログ（実験 4-1）

実験 4-2：提案指標に基づく自律的な性能管理

第 4.2 節で述べたとおり，ネットワークシステムの性能管理は，過負荷状態の回避だけでなく，利用者需要の大きさに対して計算資源の量が過剰とならないように調整することも含まれる．そこで，提案指標に基づく自律的な性能管理の実現を検証するため，提案指標を活用し，利用者需要の大きさに対して計算資源の量が過剰である状態を検出し，アラームを生成する機能を試作した．試作した本機能，そして，上述のネットワークシステムが過負荷状態に陥りつつある状態を検出する機能を併せて実験用ネットワークシステムに適用し，アラームに基づいてサービスに与える計算資源の量を調整しながらサービスを安定的に維持できるか否かを検証する．本実験で使用する，簡易なアラーム生成の手続きを以下に示す．

[時刻 p におけるアラーム生成の手続き]

- 時刻 p において，ネットワークシステムの活動度のゆらぎの分散の算出結果は，配列として $\{var(0), var(1), \ldots, var(p-1), var(p)\}$ で与えられる．
- ゆらぎの分散の算出値について，一つ前のインターバルにて算出された値との差分 d_p は，$d_p \leftarrow var(p) - var(p-1)$ で与えられる．
- 時刻 p において，単位時間あたりのリクエスト数は，配列として $\{req(0), req(1), \ldots, req(p-1), req(p)\}$ で与えられる．
- $alarm\_log$ は可変長のリストであり，過負荷状態の予兆を検出した際の単位時間あたりのリクエスト数を保持する．また，$alarm\_log[-1]$ はリストの末尾の値を指し，Append() はリストの末尾に値を追加する操作，Pop() はリストの末尾の値を削除する操作である．
- $H(H > 0)$ および $C(C > 0)$ は，それぞれゆらぎの分散のピーク値，ピーク値の超過回数に関する既定の閾値である．

- $W(W > 0)$ は，アラーム生成に際して考慮する時間幅を表すウィンドウサイズである．
- $U(U > 0)$ は，単位時間あたりのリクエスト数低下の判定に関する規定の値である．
- S_p は，時刻 $p-1$ において算出されたゆらぎの分散の値がピーク値であり，かつ，既定の閾値を超えているか否かを保持する変数である．
- 各変数は初期状態において，$p = 0$，$d_0 \leftarrow 0$，$alarm\_log = []$，$req(t) \leftarrow 0(-W+1 \geq t \geq 0)$，$S_t \leftarrow 0(-W+1 \geq t \geq 0)$ であり，各時間において以下の手続きによりアラーム生成に関する判定を行う．
- **if** $d_p \geq 0$

  then $S_p \leftarrow 0$

  else{

  if$d_{p-1} \geq 0$ **and** $var(p-1) \geq H$

  then{$S_p \leftarrow 1$

  if Sum$(S_{p-W+1}, \ldots, S_p) \geq C$

  then “Generate Alarm (SCALE-UP)”; Append($alarm\_log$,$req(p)$)

  }

  else $S_p \leftarrow 0$}

  }

  if Length($alarm\_log$) > 0

  then{

  if $req(p) \leq U * alarm\_log[-1]$ **and** Sum$(S_{p-W+1}, \ldots, S_p) = 0$

  then “Generate Alarm (SCALE-DOWN)”; Pop($alarm\_log$)

  }

表 4.2: ネットワークシステムにかける負荷の設定（実験 4-2）

Time Period [sec]	Number of Request [req/sec]
0–299	10
300–599	12
600–899	14
900–1199	16
1200–1499	14
1500–1799	12
1800–2099	10
2100–2399	12
2400–2699	14
2700–2999	16

本実験では，$H = 200$，$C = 5$，$W = 180$，$U = 0.85$ とした．これは，ゆらぎの分散に関して，180s の期間に大きさ 200 を超えるピークが 5 回以上観測された場合に計算資源の追加（スケールアップ）に関するアラームを生成するような設定である．また，ゆらぎの分散に関して，180s の期間に大きさ 200 を超えるピークが 1 回も観測されず，かつ，単位時間あたりのリクエスト数が，前回スケールアップに関するアラームを生成した際の 85%以下となった場合に，計算資源の量を減少させる（スケールダウン）ためのアラームを生成するような設定である．表 4.2 に，本実験における負荷の大きさの設定を示す．本実験では実験 4-1 と異なり負荷は増大し続けず，平均 16 リクエスト毎秒となった後に減少し，実験開始時と同様の大きさに戻った後，再び増大する．よって，効果的な性能管理のためには，ネットワークシステムの状態を見極め，適宜 VM を接続・離脱させる必要がある．

実験結果を図 4.17，図 4.18，図 4.19，図 4.20 に示す．図 4.17 はネットワークシステムに与えた負荷（リクエスト数）の推移，図 4.18 は transfer rate の計測値，また，図 4.19 は transfer rate のゆらぎの分散，そして図 4.20 はアラーム生成ログ

である．

図 4.18 に示すとおり，本実験では transfer rate に破局的な低下は発生しなかった．また，図 4.20 に示すとおり，900s 付近でスケールアップに関するアラーム，1500s 付近にスケールダウンに関するアラーム，そして 2800s 付近で再びスケールアップに関するアラームが生成されたことが分かる．図 4.18 の 900s，1600s，2800s 付近にみられる transfer rate の大幅な変化は，アラームを受けてスケールアップおよびスケールダウンを行ったことで，ネットワークシステムの処理能力が変化したことが原因である．

図 4.19 より，およそ 600–900s，2400–2800s の期間に，ゆらぎの分散が増大傾向にあり，これによってスケールアップに関するアラームが発生したことが分かる．また，スケールダウンに関するアラームが発生したおよそ 1600s 時点の直前 180s の期間では，ゆらぎの分散が収束していたことも確認できる．最初にスケールアップに関するアラームが発生した 900s 付近での負荷の設定は平均 14–16 リクエスト毎秒である一方，スケールダウンに関するアラームが発生した 1600s 付近での負荷の設定は平均 12 リクエスト毎秒であった．この結果より，本実験で試作した機能が利用者需要の大きさに対して計算資源の量が過剰である状態を捕捉可能であることを示した．

以上の結果より，提案指標は過負荷状態の予兆の検出だけでなく，サービスに与える計算資源の量が過剰となる状態の検出にも応用可能であることを示した．図 4.1 に示すとおり，効果的な性能管理の実現には，安定的なサービスの維持に必要十分な計算資源の量を見極めることが重要であった．本実験の結果より，提案指標に基づいて性能管理に関する管理知識を記述することで，ネットワークシステムの動作特性に関する詳細な事前知識が無くとも，ネットワークシステムの運用状態を的確に捉え，安定的なサービス維持のための計算資源の量の調整が可能である事を示した．

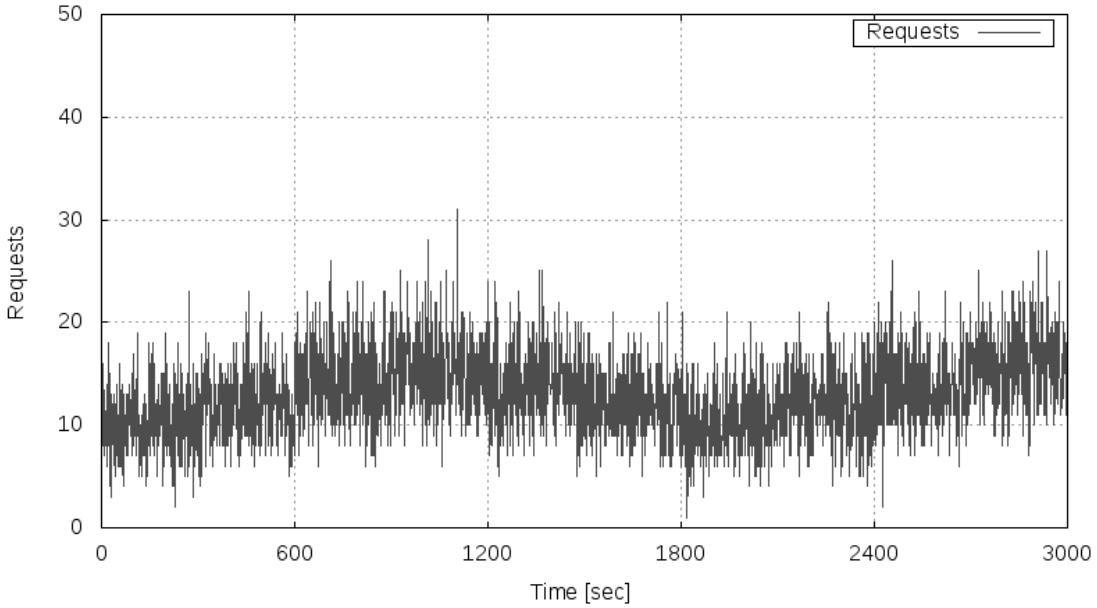


図 4.17: 負荷の推移（実験 4-2）

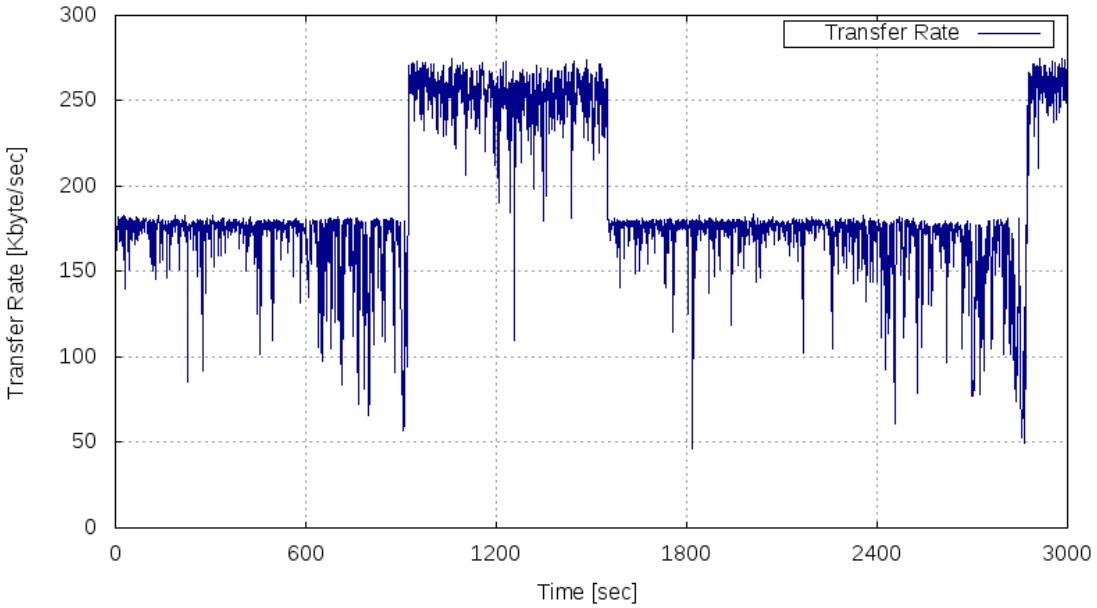


図 4.18: transfer rate の計測値（実験 4-2）

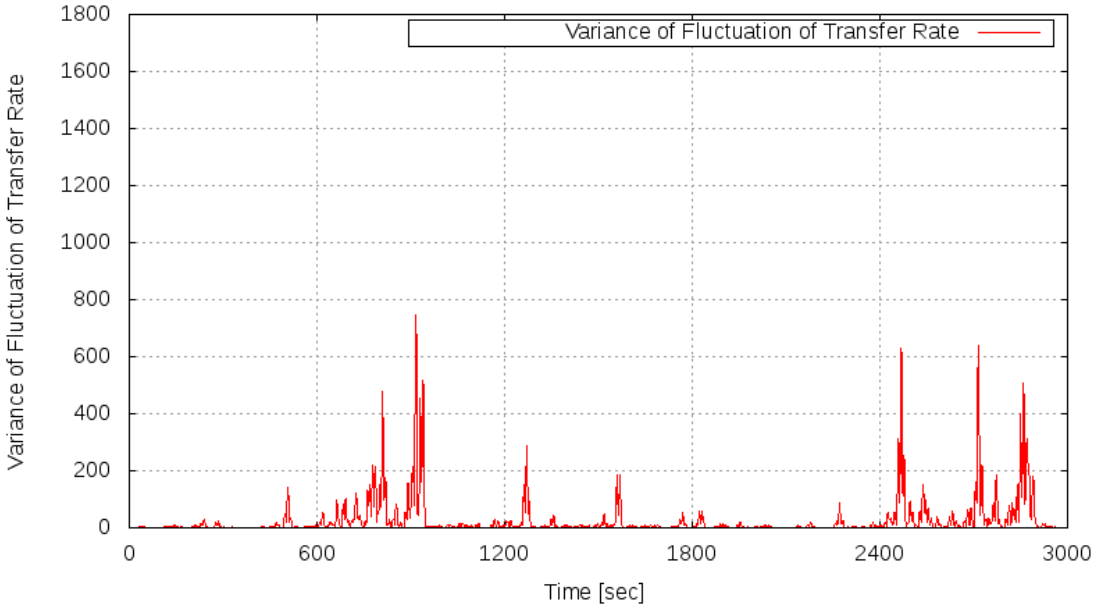


図 4.19: transfer rate のゆらぎの分散（実験 4-2）

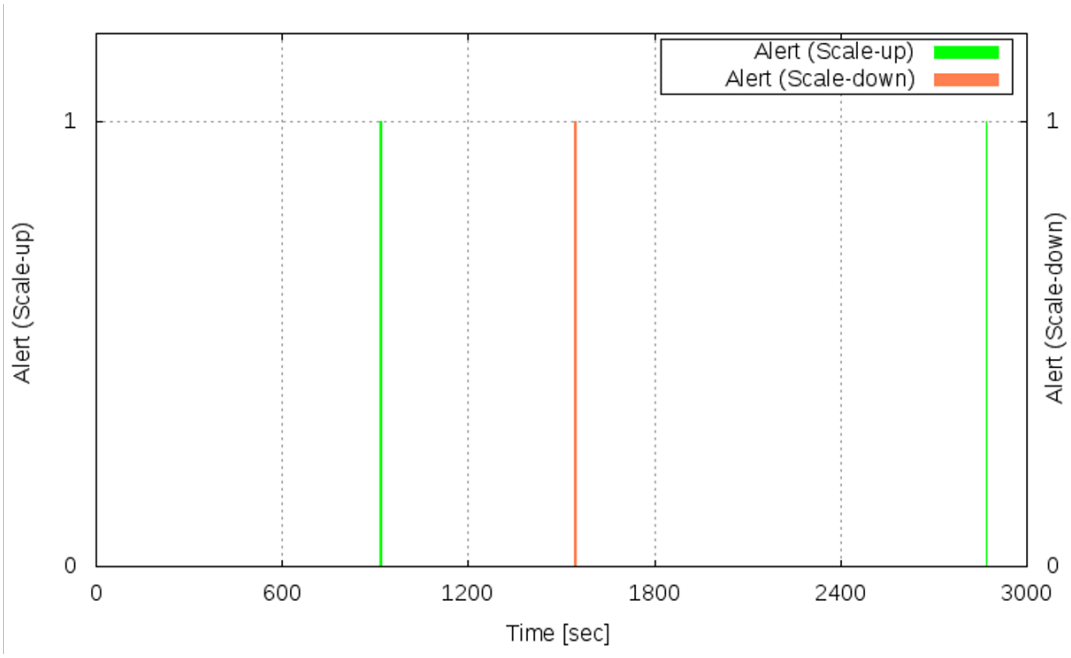


図 4.20: アラーム生成ログ（実験 4-2）

4.4.4　評価

実験 3 では提案に示す作業仮説の検証を行った．実験結果より，提案指標，すなわち，ネットワークシステムの活動度のゆらぎの分散が，提案に示す分析結果どおりにふるまうことを確認した．これより，ネットワークシステムの活動度のゆらぎの分散というマクロな観点に基づく提案指標を観測することで，システムの運用状態が把握できることを示した．

実験 4 では，実験 3 を通して検証された提案指標に基づき，自律的な性能管理が可能であるかどうか，その適用可能性を検証した．実験結果より，提案指標を組み込んだ実験システムでは，ネットワークシステムの状態を捉え，的確なタイミングでスケーリングを施すことにより，過負荷状態，および計算資源の量が過剰な状態を回避しながら安定的にサービスを維持できることを示した．

以上より，提案（S2）巨視的な観点に基づくネットワークシステムの内部状態の観測法 により，課題（P2）動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 が解決可能であることが示された．ゆえに，本提案に基づく KNMS は，変動するネットワークシステムにおいても適用可能であり，その自律的な性能管理の実現に貢献可能であると言える．

4.5　まとめ

本章では，変動するネットワークシステムに適用可能な KNMS の実現に向け，特に自律的な性能管理に焦点を当て，2 章にて述べた課題（P2）の解決を目指し，（S2）巨視的な観点に基づくネットワークシステムの内部状態の観測法 を提案した．実験結果より，提案（S2）により課題（P2）が解決可能であることを示し，変動するネットワークシステムにおいても自律的な性能管理が可能な KNMS の実現可能性を示した．

第5章　結論

第5章「結論」では，本論文のまとめと本研究の貢献について述べ，今後に残された課題について述べる．

5.1　各章のまとめ

第1章「序論」では，本論文の研究背景について概説し，本研究の対象であるKNMSを用いた自律的なネットワーク運用について述べ，既存のKNMSに関する課題の概要を述べた．そして，本研究の目的と焦点，既存研究に対する本研究の位置づけを示した．

第2章「知識型ネットワーク管理システムの現状と課題」では，KNMSに関する研究を取り上げ，管理知識の利活用に関する課題を詳細化した．特に，KNMSに基づくネットワーク運用の自律化に関して，障害管理の自律化，そして，性能管理の自律化に焦点を当て，それぞれに関して具体的な課題を述べた．KNMSによる自律的な障害管理に関しては，(P1) 知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難 という課題があり，KNMSによる自律的な性能管理に関しては，(P2) 動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 という課題があった．そして，これらの課題の解決に向けたアプローチを概説した．

第3章「知識型ネットワーク管理システムによる自律的な障害管理」では，自律的な障害管理の実現を目指すKNMSに関する課題である，(P1) 知識ベース内の知識構成を熟知していなければ，新たな管理知識の追加や変更が困難 の解決を目

指し,（S1）モジュール化に基づくサービス指向型の知識管理法 を提案した．本提案では，管理知識を任意のまとまりでモジュール化することで，システム構成の変化の原因となったサービスに焦点を当てた知識ベースの更新を可能とした．これにより，知識ベース全体の知識構成を熟知していない管理者であっても容易に知識ベースを更新することが可能となった．また，本提案に基づいて試作システムを設計・実装し，既存システムとの比較実験を行った．実験結果より，提案に基づく KNMS は知識ベース更新にかかる管理者負担が大きく軽減されたことを確認した．これより，提案（S1）によって課題（P1）が解決可能であることを示し，KNMS を用いた自律的な障害管理の実現に本提案が貢献可能であることを示した．

第 4 章「知識型ネットワーク管理システムによる自律的な性能管理」では，自律的な性能管理を目指す KNMS に関する課題である,（P2）動作特性に関する事前知識がなければ，ネットワークシステムの運用状態を把握することは困難 の解決を目指し,（S2）巨視的な観点に基づくネットワークシステムの内部状態の観測法を提案した．本提案では，ネットワークシステムの活動度のゆらぎの分散というマクロな観点に基づく指標を観測することで，構成の変動によって変化する，ネットワークシステムの詳細な動作特性を用いずとも，ネットワークシステムの運用状態，特に，ネットワークシステムが不都合な状態に陥りつつあることを把握可能とした．さらに，本提案を検証するため，実機を用いて実験用システムを構築し評価実験を行った．実験結果より，本指標が，提案に示すネットワークシステムの動作モデルに関する解析結果と同様に振る舞うことを確認した．また，提案指標に基づいて自律的な性能管理が可能であるかどうか，その適用可能性を検証し，実験結果より動作特性に関する事前知識を用いずに，ネットワークシステムの運用状態を的確に捉え，安定的にサービスを維持可能であることを示した．これより，提案（S2）によって課題（P2）が解決可能であることを示し，KNMS を用いた自律的な性能管理に本提案が貢献可能であることを示した．

5.2　本研究の成果

本研究では従来からのネットワーク管理者の人材不足，そして今後の ICT 技術・ICT サービスのさらなる普及と進展に向けて，KNMS によるネットワークシステムの自律的な運用に焦点を当てた．従来の KNMS に関する取り組みは特定の環境，すなわち，KNMS の自律性を司る管理知識を頻繁に追加・変更する必要の無い，構成の変化しないネットワークシステム等を対象とするなど，限定的な適用範囲にとどまっていた．そこで本研究では，より実用的な KNMS の実現および KNMS の普及を妨げる原因として管理知識の利活用に関して課題があると考え，課題の解決を図った．

障害管理の自律化を目指す KNMS は，その管理知識として，発生した障害を分析し原因を特定する知識，そして，障害から回復するためにとるべき行動を定める知識を，知識ベースに蓄積する．このような KNMS が正常に動作するためには知識ベース内の知識の整合性がとれていることが前提であるため，知識ベースに新たな知識を追加したり，知識ベース内にある知識に修正を加えたりする作業は，知識ベース内の知識構成に関して細心の注意を払う必要があり，容易に行えるものではない．また，KNMS の運用の過程で知識ベースが肥大化した場合，知識の追加・変更はさらに難しいものとなる．そこで本研究では管理知識をモジュール化し，任意のまとまりで管理可能とすることで，知識ベースの更新にかかる管理者負担の軽減を図った．さらに，複数のモジュール間で限定的に管理知識を連携させる機能を与えることで，管理対象システムをそれぞれ独立したサービスとして切り分けることが難しい場合においても各サービスに焦点を当てた知識記述を可能とし，モジュール化の利点と管理者支援システムとしての実用性の両立を図った．以上の成果は，KNMS を用いた自律的な障害管理における管理知識の新しい利用手法として評価できる．さらに，本提案の有効性を示すため，提案手法を適

用したプロトタイプシステムを設計・実装し，評価実験を行ったが，これはエージェント型ネットワーク管理システムの先導的な設計事例としても有用である．

そして，性能管理の自律化を目指すKNMSは，その管理知識として，サービスの運用状態を捉えるための知識，そして，サービスの安定状態を保つためにとるべき行動を定める知識を，知識ベースに持つ．ネットワークシステムは高負荷時において非線形的な動態を有するため，サービスの運用状態について，特に正常状態から過負荷状態へ遷移するタイミングを捉えるためには，事前にネットワークシステムの動作特性を詳細に測定しておく必要がある．しかし，ネットワークシステムの動作特性はハードウェア的・ソフトウェア的な構成により多様に変化するため，変化の都度，測定し直す必要があり，詳細な動作特性に基づいてシステムの運用状態を捉える管理知識は汎用性が低い．また，動作特性の測定にかかる手間と時間を考慮すると，管理知識の更新は容易に行えるものでは無い．そこで本研究ではネットワークシステムの挙動をマクロな観点から観測することで得られる，ネットワークシステムのマクロな挙動を示す値を活動度と定義し，活動度に含まれるゆらぎの性質に着目し，分析を行った．分析結果より，ネットワークシステムの活動度のゆらぎの分散がその運用状態を捉えるための指標となることを示し，ネットワークシステムの詳細な構成および動作特性に依存しない，汎用的な管理知識の記述が可能であることを示した．以上の成果は，ネットワークシステムにおける自律的な性能管理の実現に向けた独創的な手法として評価できる．

上述の通り，本研究では管理知識の利活用に関する課題を解決することで，KNMSの管理性を改善し，その適用範囲を拡げることができた．すなわち本論文は，ネットワークシステムの自律的な運用管理に向けた知識型ネットワーク管理システムの実現に係る新しい手法を提案したもので，ネットワーク管理運用技術，ならびに，情報基礎科学の発展に寄与するところが少なくない．

5.3　今後の課題

今後の課題として，知識型ネットワーク管理システムによる自律的な障害管理に関しては，モジュール間の管理知識の連携を司る Facilitator-Agent のさらなる高度化が挙げられる．評価実験で用いた試作システムでは，サービス間の依存関係に関する記述を管理知識（Ksc）に含める必要があった．モジュール間の知識連携の事前検証機能をより高度化し，モジュールの独立性をさらに高めることで，知識の整合性確認に関する管理者負担はさらに軽減されると考えられる．

知識型ネットワーク管理システムによる自律的な性能管理に関しては，機器の故障等の内的要因に起因する過負荷状態の検出が挙げられる．ネットワークシステムが過負荷状態に陥る原因として，サービス要求の増大などの外的要因の他，ネットワークシステム内のハードウェアの故障，あるいは，ソフトウェアのバグなど，内的要因によるシステムの計算能力の低下も考えられる．提案指標を用いた内的要因による運用状態の変化の検出を検証するためには，より規模の大きな実験システムを構築する必要がある．これに関連して，提案指標の障害管理分野への応用も挙げられる．

また，本研究では主にネットワークシステムの運用フェーズにおける管理者支援に焦点を当てたが，ネットワークシステムの構築フェーズにおける管理者支援，すなわち，ネットワークサービスの自動構築技術との連携も挙げられる．これにより，ネットワークシステムのライフサイクルにおける，より包括的な管理者支援の実現が期待される．

謝辞

末筆ながら，本研究を行うにあたり，多くの方々よりご指導・ご助言をいただきました．ここに，心より深く感謝の意を表します．

特に，本研究を行うにあたり，日頃より懇切なご指導を賜りました，東北大学電気通信研究所教授 木下哲男先生に心から感謝いたします．木下哲男先生には，学部生の頃から，長きにわたりお世話になりました．また，論文の執筆に際しては辛抱強く励ましてくださいました．心よりお礼申し上げます．

東北大学電気通信研究所教授 鈴木陽一 先生，そして，東北大学大学院情報科学研究科教授 篠原歩先生には，本論文の審査にあたり，多くの有益なご指導とご助言を賜りましたことに深く感謝いたします．

また，審査時や日頃の研究活動を支えていただきました，東北大学電気通信研究所准教授 北形元先生に心より感謝いたします．

さらに，東北大学電気通信研究所助教 笹井一人先生には本研究を遂行するにあたり数多くのご助言やご指導をいただきました．ここに感謝申し上げます．そして，東北大学電気通信研究所助教 高橋秀幸先生をはじめとする，木下研究室のスタッフ・学生・卒業生の皆様には，ゼミ等における討論や研究室生活をおくる上で大変お世話になりました．心からお礼申し上げます．

最後に，長きにわたる学生生活を温かく見守りそして支えてくれた両親に深く感謝し，本論文を締めくくります．

参考文献

[1] 総務省，“平成 29 年版 情報通信白書,” [Online]. http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/index.html.

[2] 総務省，“平成 28 年版 情報通信白書,” [Online]. http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/index.html.

[3] Cisco, “Embracing the internet of everything to capture your share of $14.4 trillion,” [Online]. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf, 2013.

[4] 総務省，“平成 28 年通信利用動向調査,” [Online]. http://www.soumu.go.jp/johotsusintokei/statistics/data/170608_1.pdf.

[5] Nagios, [Online]. https://www.nagios.org/.

[6] Zabbix, [Online]. https://www.zabbix.com/.

[7] Sensu - Full-stack monitoring for today’s business., [Online]. https://sensuapp.org/.

[8] J.O. Kephart and D.M. Chess, “The vision of autonomic computing,” Computer, vol.36, no.1, pp.41–50, 2003.

[9] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A survey of autonomic network architectures and evaluation criteria," IEEE Communications Surveys & Tutorials, vol.14, no.2, pp.464–490, 2012.

[10] E.C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," IEEE Transactions on Software Engineering, vol.25, no.6, pp.852–869, 1999.

[11] N. Samaan and A. Karmouch, "Towards autonomic network management: an analysis of current and future research directions," IEEE Communications Surveys & Tutorials, vol.11, no.3, pp.22–36, 2009.

[12] L. Raman, "OSI systems and network management," IEEE Communications Magazine, vol.36, no.3, pp.46–53, 1998.

[13] M. Ross, A. Covo, and C. Hart, "An AI-based network management system," Computers and Communications, 1988. Conference Proceedings., Seventh Annual International Phoenix Conference on, pp.458–461, 1988.

[14] R.N. Cronk, P.H. Callahan, and L. Bernstein, "Rule-based expert systems for network management and operations: an introduction," IEEE Network, vol.2, no.5, pp.7–21, 1988.

[15] J. Keeney, S. van derMeer, and G. Hogan, "A recommender-system for telecommunications network management actions," Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on, pp.760–763, 2013.

[16] S. Montani and C. Anglano, "Case-based reasoning for autonomous service failure diagnosis and remediation in software systems," European Conference on Case-Based Reasoning, pp.489–503, 2006.

[17] G. Tesauro, D.M. Chess, W.E. Walsh, R. Das, A. Segal, I. Whalley, J.O. Kephart, and S.R. White, "A multi-agent systems approach to autonomic computing," Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, pp.464–471, AAMAS '04, 2004.

[18] H.M. Tran and J. Schönwälder, "Distributed case-based reasoning for fault management," IFIP International Conference on Autonomous Infrastructure, Management and Security, pp.200–203, 2007.

[19] F. Zaman, G. Hogan, S. Van Der Meer, J. Keeney, S. Robitzsch, and G.-M. Muntean, "A recommender system architecture for predictive telecom network management," IEEE Communications Magazine, vol.53, no.1, pp.286–293, 2015.

[20] K. Sasai, J. Sveholm, G. Kitagata, and T. Kinoshita, "A practical design and implementation of active information resource based network management system," International Journal of Energy, Information and Communications, vol.2, no.4, pp.67–86, 2011.

[21] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "Vmware distributed resource management: Design, implementation and lessons learned," VMware Technical Journal, vol.1, no.1, pp.45–64, 2012.

[22] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," ACM SIGOPS operating systems review, vol.37, pp.164–177, 2003.

[23] Microsoft Hyper-V Server, [Online]. https://www.microsoft.com/en-us/cloud-platform/server-virtualization.

[24] A. Cockcroft, "Utilization is virtually useless as a metric!," Annual International Conference of the Computer Measurement Group 2006 (CMG2006), pp.612–617, 2006.

[25] A.R. Hummaida, N.W. Paton, and R. Sakellariou, "Adaptation in cloud resource configuration: A survey," Journal of Cloud Computing, vol.5, no.1, pp.57:1–57:16, 2016.

[26] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "Qos–aware clouds," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pp.321–328, 2010.

[27] S. Singh and I. Chana, "QoS-aware autonomic resource management in cloud computing: a systematic review," ACM Computing Surveys (CSUR), vol.48, no.3, pp.42:1–42:46, 2016.

[28] R.H. Hwang, C.N. Lee, Y.R. Chen, and D.J. Zhang-Jian, "Cost optimization of elasticity cloud resource subscription policy," IEEE Transactions on Services Computing, vol.7, no.4, pp.561–574, 2014.

[29] D. Dib, N. Parlavantzas, and C. Morin, "SLA-based profit optimization in cloud bursting PaaS," Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pp.141–150, 2014.

[30] Z. Li and G. Wu, "Optimizing VM live migration strategy based on migration time cost modeling," Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems, pp.99–109, 2016.

[31] D. Mosberger and T. Jin, "httperf – a tool for measuring web server performance," ACM SIGMETRICS Performance Evaluation Review, vol.26, no.3, pp.31–37, 1998.

[32] G. Banga and P. Druschel, "Measuring the capacity of a web server under realistic loads," World Wide Web, vol.2, no.1-2, pp.69–83, 1999.

[33] J. Dilley, R. Friedrich, T. Jin, and J. Rolia, "Web server performance measurement and modeling techniques," Performance Evaluation, vol.33, no.1, pp.5–26, 1998.

[34] T.F. Abdelzaher and Chenyang Lu, "Modeling and performance control of Internet servers," Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187), vol.3, pp.2234–2239, 2000.

[35] J. Cao, M. Andersson, C. Nyberg, and M. Kihl, "Web server performance modeling using an M/G/1/K*PS queue," 10th International Conference on Telecommunications, 2003. ICT 2003., vol.2, pp.1501–1506, 2003.

[36] C. Stewart and K. Shen, "Performance modeling and system management for multi-component online services," Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2, pp.71–84, 2005.

[37] D.C. Verma, Principles of computer systems and network management, Springer, 2009.

[38] R.G. Garroppo, S. Giordano, S. Spagna, and S. Niccolini, "Queueing strategies for local overload control in SIP server," Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE, pp.1–6, 2009.

[39] C. Poellabauer, A. Schwan, and R. West, "Lightweight kernel/user communication for real-time and multimedia applications," Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video, pp.145–154, 2001.

[40] T. Sakano, Z.M. Fadlullah, Thuan Ngo, H. Nishiyama, M. Nakazawa, F. Adachi, N. Kato, A. Takahara, T. Kumagai, H. Kasahara, and S. Kurihara, "Disaster-resilient networking: a new vision based on movable and deployable resource units," IEEE Network, vol.27, no.4, pp.40–46, 2013.

[41] Movable and Deployable ICT Resource UNIT, [Online]. http://www.mdru.org/.

[42] T. Kinoshita, "Agent-based active information resource and its applications," Databases in Networked Information Systems, pp.143–156, Springer, 2010.

[43] Y. Takahashi, D. Misugi, A. Sakatoku, A. Satoh, A. Takahashi, K. Sasai, G. Kitagata, T. Abe, and T. Kinoshita, "Knowledge oriented network fault resolution method based on active information resource," Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on, vol.2, pp.361–364, 2010.

[44] Y. Tanimura, J. Sveholm, K. Sasai, G. Kitagata, and T. Kinoshita, "A knowledge-based support method for autonomous service operations after dis-

asters," Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on, pp.229–233, 2013.

[45] T. Uchiya, H. Hara, K. Sugawara, and T. Kinoshita, "Repository-based multiagent framework for developing agent systems," Transdisciplinary Advancements in Cognitive Mechanisms and Human Information Processing, pp.60–79, 2011.

[46] IDEA, [Online]. http://www.k.riec.tohoku.ac.jp/s/idea/index.html.

[47] P. Martin, A. Brown, W. Powley, and J.L. Vazquez-Poletti, "Autonomic management of elastic services in the cloud," Computers and Communications (ISCC), 2011 IEEE Symposium on, pp.135–140, 2011.

[48] J. Kirschnick, J.M. Alcaraz Calero, P. Goldsack, A. Farrell, J. Guijarro, S. Loughran, N. Edwards, and L. Wilcock, "Towards an architecture for deploying elastic services in the cloud," Software: Practice and Experience, vol.42, no.4, pp.395–408, 2012.

[49] I.K. Kim, J. Steele, Y. Qi, and M. Humphrey, "Comprehensive elastic resource management to ensure predictable performance for scientific applications on public iaas clouds," Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp.355–362, 2014.

[50] T. Kinoshita, "Basic characteristics of a macroscopic measure for detecting abnormal changes in a multiagent system," Sensors, vol.15, no.4, pp.9112–9135, 2015.

[51] K.M. Elleithy and A. Komaralingam, "Using a queuing model to analyze the performance of web servers". International Conference on Advances in

Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet, 2002.

[52] M.R. Moghal, N. Hussain, M.S. Mirza, M.W. Jarral, and M.S. Choudry, "Performance evaluation and modeling of web server systems," 8th WSEAS International Conference on Computers, pp.9–11, 2004.

[53] M.C. Wang and G.E. Uhlenbeck, "On the theory of the Brownian motion II," Reviews of Modern Physics, vol.17, no.2-3, pp.323–342, 1945.

[54] L.M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," ACM SIGCOMM Computer Communication Review, vol.41, no.1, pp.45–52, 2011.

[55] ab - Apache HTTP server benchmarking tool, [Online]. https://httpd.apache.org/docs/2.4/programs/ab.html.

発表論文

Journal papers

1. Kazuto Sasai, Yusuke Tanimura, Hideyuki Takahashi, Gen Kitagata, Tetsuo Kinoshita, "An Agent-based Data Analytics Support Tool for Network Management Intelligence," International Journal of Energy, Information and Communications, Vol. 8, No. 1, pp. 51–64, Feb. 2017.

2. Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, "Knowledge-Based Network Management System for Movable and Deployable ICT Resource Unit," Journal of Computer and Communications, Vol.5, No.7, pp.135–151, May 2017.

3. Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, "Analysis of the Macroscopic Behavior of Server Systems in the Internet Environment," Applied Sciences, Vol.7, No.11, pp.1145:1–1145:22, Nov. 2017.

4. Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, "Knowledge-Based Network Management Support for Preemptive Service Management," (In Preparation)

Conference papers

5. Yusuke Tanimura, Johan Sveholm, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, "A Knowledge-based Support Method for Autonomous Service Operations after Disasters," Proc. of the 12th IEEE/ACIS International Conference on Computer and Information Science (ICIS2013), pp.229–233, Jun. 2013.

6. Yusuke Tanimura, Johan Sveholm, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, “An Autonomic Network Service Management using Knowledge-based Support System,” Proc. of the 2nd International Workshop on Smart Technologies for Energy, Information and Communication (STEIC2013), pp.96–104, Aug. 2013.

7. Johan Sveholm, Khamisi Kalegele, Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, “A Knowledge-based Autonomous Service Management System in Emergency Situations,” Proc. of the 5th IEEE International Conference on Awareness Science and Technology (iCAST2013), pp.109–113, Oct. 2013.

8. Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, “Service Oriented Network Management with Knowledge-Based Network Management System in Fluctuating Environment,” Proc. of the 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017), pp.31–44, June 2017.

9. Takuya Hoshino, Yusuke Tanimura, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, “Support Mechanism for the Collaboration between Humans and Agents in Network Management Tasks,” Proc. of the 6th IEEE Global Conference on Consumer Electronics (GCCE 2017), pp.399–400, Oct. 2017.

国内研究会・ワークショップ論文集

10. 谷村 優介, Sveholm Johan, 笹井 一人, 北形 元, 木下 哲男, “災害時の迅速な障害復旧のための知識型管理支援に関する研究,” 情報処理学会東北支部研究報告, Vol2012-4, 2012-2-3, pp.1–5, Jan. 2013.

11. 谷村 優介, Sveholm Johan, 笹井 一人, 北形 元, 木下 哲男, “耐災害 ICT ユニットのための知識型管理運用支援方式の提案,” 情報処理学会第 75 回全国大会講演論文集, 5X-5, pp.3-371–3-372, Mar. 2013. (学生奨励賞)

12. 谷村 優介, Sveholm Johan, 笹井 一人, 北形 元, 木下 哲男, “災害時の応急的ネットワークサービスの知識型運用支援システム,” 第 4 回先進的情報通信工学研究会, Mar. 2013.

13. 谷村 優介, 笹井 一人, 北形 元, 木下 哲男, “AIR に基づくネットワークサービスの自律的管理支援システム,” 2014 年電子情報通信学会総合大会通信講演論文集 1, B-15-17, p.570, Mar. 2014.

14. 谷村 優介, 笹井 一人, 北形 元, 木下 哲男, “能動的情報資源に基づく応急的ネットワークサービスの管理運用支援,” モバイルネットワークとアプリケーション研究会（MoNA）, pp.19–24, Sep. 2014.

15. 中橋 修平，谷村 優介，笹井 一人，北形 元，木下 哲男, “知識型ネットワーク管理システムにおける異常検知機能の開発,” 第 10 回先進的情報通信工学研究会, Nov. 2014.

16. 谷村 優介, 笹井 一人, 北形 元, 木下 哲男, “動的に変化するネットワークシステムのための知識型障害解決支援システム,” 第 23 回マルチメディア通信と分散処理ワークショップ (DPSWS2015) 論文集, pp.156–163, Oct. 2015. (優秀論文賞・情報処理学会 2016 年度 山下記念研究賞)

17. 山田 良介, 谷村 優介, 笹井 一人, 高橋 秋典, 北形 元, 五十嵐 隆治, 木下 哲男, “エージェントに基づくネットワークデータ分析支援基盤,” 情報処理学会第 78 回全国大会講演論文集, 5R-01, pp.3-73–3-74, Mar. 2016.

18. 松村 洋志，山田 良介，谷村 優介，笹井 一人，北形 元，木下 哲男, “情報流処理基盤のためのデータ取得点のエージェント化に関する研究,” 平成 28 年度電気関係学会東北支部連合大会講演論文集, p.1D10, Aug. 2016.

19. 星野 拓也，木下 哲男，北形 元，笹井 一人，谷村 優介, “複数の人とエージェントの協働を支援するメッセージ共有機構の開発,” 第 1 8 回先進的情報通信工学研究会, Feb. 2017.

20. 星野 拓也，谷村 優介，笹井 一人，北形 元，木下 哲男, “ネットワーク管理におけるコレクティブインタラクション支援機構の検討,” 情報処理学会第 78 回全国大会講演論文集, 4K-04, pp.1-483–1-484, Mar. 2017.

21. 谷村 優介，笹井 一人，北形 元，木下 哲男, “マクロな動作状況に着目したWeb サーバの異常状態の予測と制御,” 情報処理学会第 78 回全国大会講演論文集, 3D-04, pp.3-53–3-54, Mar. 2017.