耐障害性・応答性向上のためのモバイルWebサービスプラットフォーム

吉川 貴　太田 賢　中川 智尋　倉掛 正治
NTTドコモ マルチメディア研究所

本研究では、モバイル環境での分散アプリケーション利用について安定したサービス品質の提供を目的とし、迅速な障害検知や代替リソースへの切り替えによるサービス回復を行う、モバイルWebサービスプラットフォームを検討する。提案するプラットフォームは、クライアントアプリケーション単位での細かいサービス品質要求を、端末内のミドルウェアとオーバレイネットワーク型のプラットフォームの協調で実現する。本報告では、アプリケーションの要求に合わせて迅速かつネットワーク負荷を抑えた障害検知を行う適応的モニタリングと、障害の通知に係るネットワーク負荷を抑えるモニタリング情報通知プロトコルについて提案する。また、インターネット上の実サービスを含むテストベッドで実験を行い、提案手法の機能や性能、有効性について評価を行った。

Mobile Web Service Platform for Robust, Responsive Distributed Application

Takashi Yoshikawa　Ken Ohta　Tomohiro Nakagawa　Shoji Kurakake
Multimedia Laboratories, NTT DoCoMo, Inc.

In mobile computing environments, distributed applications are established over unstable wireless network. The goal of our research is to offer stability of service by rapid failure detection and switching to alternative service. Adaptive Monitoring and Monitoring Information Notification Protocol, which enable both rapid failure detection and little network load are proposed. We confirmed that our systems detected failure and switched to alternative service in recovery time that required by application, as a result of experiments with a testbed that contains a real web services in the Internet. It is also confirmed that proposed method is effective to reduce network load as well as satisfy requirements of application.

1. はじめに

昨今、セルラー通信や無線 LAN、短距離無線通信といった無線通信技術の発展にともない、ユーザの移動を束縛しない自由なモバイルコンピューティング環境が実現されている。一方、インターネットにおいて、Web サービスや CDN、P2P プラットフォームなどの広域分散サービスが注目を集め、その開発が進んでいる。

しかし、モバイルコンピューティング環境では、ユーザの移動や不安定な無線リンク、サーバの負荷や障害などにより、安定した品質でサービスを提供することは困難である。高品質で安定したサービスをユーザに提供し、複雑かつ動的な環境の変化をユーザやアプリケーションから隠蔽するためには、不安定なネットワークやリソース状態を自律的に監視し、障害に対して動的に適応できる基盤を構築する必要がある。

特にモバイル環境ではユーザの状況や使用するアプリケーションが多岐に渡り、高い QoS 保証が重要となることも少なくない。例えば、駅で電車の乗り換え案内サービスを利用する場合、検索に時間がかかってしまっては電車を逃してしまうかもしれないし、一分一秒を争う株取引のサービスには、非常に高い応答性が要求されると考えられる。そこで本研究では、様々な QoS 制御に関するトピックの中でも、特にアプリケーションレベル QoS の応答性の保証について焦点をあてて検討を進める。

本稿では、高い応答性を保証するために、迅速な障害検知と代替サービスへの切り替えをプラットフォームの機能として提供する。また、障害検知の迅速性とネットワーク負荷の低減の両立を課題とし、適応的モニタリング及びモニタリング情報通知プロトコルについて提案を行う。提案手法ではアプリケーションからの回復要求に応じてモニタリング頻度を変化させ、可能な限りネットワーク負荷を軽減する。

以下、2 章で応答性についての課題と関連研究をあげ、3 章では提案するモバイル Web サービスプラットフォームの詳細を述べる。4 章では評

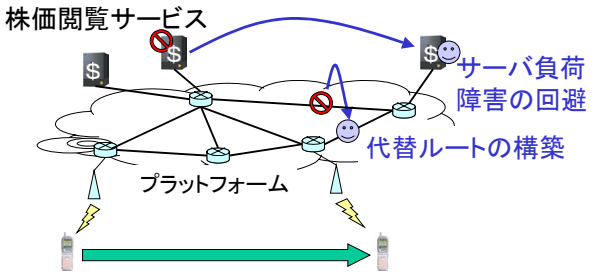

図 1 株価チェックアプリケーションの利用

価用テストベッドの実装を示し、5 章で障害検知の迅速性やネットワーク負荷の測定結果について述べる。

2. 背景

2.1 サービスの応答性

図 1 に、ユーザが移動しながら、数秒に一度の割合で定期的に株価チェックのアプリケーションを利用するシナリオを例に挙げ、現状の分散アプリケーション利用における課題を述べる。ユーザの携帯端末では株価チェックのアプリケーションプログラムが動作し、インターネット上の株価チェック Web サービスを使用して更新された株価を取得している。

ある時、Web サービスを多数のユーザが同時に使用し、サービスノードや途中のネットワーク経路に多量の負荷がかかり、応答性能が低下したとする。このような場合に、従来ではサービス提供が中断し、ユーザはサービスの回復を待ちつづけるか、手動で別のサービスに切り替えて利用する必要があった。

これに対して、提案するプラットフォームではサービスノードの性能低下を検知し、代替サービスを発見して提供することで障害から回復することを迅速に行い、サービスの中断時間を出来る限り短くする。サービスの中断時間(Tall)は、障害検知に要する時間(Ta)、代替サービスを発見する時間(Tb)、サービスの切り替えに要する時間(Tc)、サービス実行時間(Td)の総和で求められる(式 a)。そのため、迅速なサービス回復を行うためには、障害検知とサービスの復旧の双方について、検討する必要がある。

$$Tall = Ta + Tb + Tc + Td \quad \cdots(a)$$

1. 迅速な障害検知

モバイルコンピューティング環境では無線リンク部分や有線のネットワーク経路、サービスを提供するサーバで障害や性能低下が発生する。現状、障害や性能低下を検知するためには、タイムアウトによる検知やアクティブ型のモニタリング手法が多く使われるが、前者には検知の迅速性の点で問題がある。また、後者の手法では頻繁なモニタリングによって迅速な障害検知が可能になるが、そのためにネットワークに与える負荷が大きくなり、スケーラビリティの面で問題となる。このように障害から迅速に復旧してサービスの中断時間を減らすためには、できるだけ迅速な障害検知処理、ネットワーク負荷の減少が重要な課題となる。

2. 代替サービスの発見による迅速な回復

障害を検知した後に、障害発生個所に応じて適切な復旧処理を可能な限り迅速に行う必要がある。現状ではサービスが回復するまで処理を中断するか、ユーザが手動で代替サービスに切り替えて復旧を行う必要があるが、サービスの中断時間を短縮するためには、自動的に代替サービスを発見し、切り替え処理を行ってサービスの復旧を行う必要がある。また、広域に配置されている分散サービスのディスカバリの課題があり、現在では P2P 型や集中型のディスカバリシステムが提案されているが、迅速性や耐障害性、スケーラビリティを全て満たすことは困難である。

本研究は、これらの課題を解決するための機能をプラットフォームとして提供することで、サービス提供者やユーザからモバイルネットワークの不安定性を隠蔽し、耐故障性・応答性において高性能なサービス提供を可能にする。

2.2 前提条件

「同機能を提供するサービス」を代替サービスと定義し、サービスプロバイダが複数の代替サービスを用意していることや、他のプロバイダの提供する代替サービスが存在し、UDDI などのサービス検索機構で検索可能であると仮定する。

また、ステートレスなサービスの場合、単純な切り替え処理により障害から回復できるが、オンラインショッピングのようにステートを保持する必要のあるサービスに対しては、代替サービスがステート情報を同期しているものと仮定する。ステート情報の例として、オンラインショッピングの場合には、選択した商品やユーザの入力したクレジットカード番号などが挙げられる。

2.3 関連研究

オーバレイネットワーク

RON(Resilient Overlay Networks)[1]は、既存

の BGP-4 で障害の検知から通知、代替経路の発見、経路の復旧にいたるまで数分かかる処理を、ノード数を制限したオーバレイネットワークによる障害や性能情報の頻繁な交換とルーティングアルゴリズムによって十数秒で実行可能にしている。

X-Bone[2]はネットワークリソースの動的な発見やモニタリングを行い、オーバレイネットワークのコンフィギュレーションを自動的に行うシステムである。SNMP を用いてモニタリングしたネットワーク情報をオーバレイノードで管理し、ノード同士の協調動作を行うことで、障害時のネットワークトポロジの再構成を行い、障害時の自動適応を行う。

やわらかいネットワーク[3]は複数のオーバレイノードを用いてルーチングや QoS 制御を行うオーバレイネットワークである。様々な環境変化要因をネットワーク内部と、外部の環境変化に分類し、それらに対する対応策の自律的な発見・実行が検討されている。

耐障害性を向上するために、レプリケーションの手法が多く用いられる。Oceanstore[4]は大規模なインターネットに分散した P2P の共有ストレージシステムである。Oceanstore ノードにオブジェクトの複製を分散的に配置する方法と、リソース発見の際に最も近い複製にクエリをルーチングする手法に特徴がある。

以上のように、オーバレイネットワークを用いることで、既存の IP ネットワークに変更を加えずに、アプリケーションの要求に特化した障害検知やルーチング処理を実現することが可能になる。本研究では、プラットフォームの構築にオーバレイネットワークのアプローチを利用する。

リソース記述方式

分散されたコンポーネントとしてのサービスやデバイスには、その目的、処理内容、入出力インタフェースなどについて自己記述することが求められており、CC/PP[5]や WSDL[6]などの記述方式が提案されている。記述方式の課題には、記述に柔軟性や拡張性を持たせることや、単純なバイナリデータに比べて増加しがちなデータ量や計算量を低減することが挙げられる。

RDF(Resource Description Framework)[7]は Web 上のリソースの効率的な検索・発見を行うことを目的に開発されたリソースのメタデータを記述する言語である。本研究では、リソース属性やリソース間の関係性について柔軟で拡張性のある記述方式として RDF を用い、モニタリングによって得られたリソース状態を記述する。

モニタリング手法

サーバに対する負荷や障害情報の管理や通知のためのプロトコルとして SNMP(Simple Network Management Protocol)[8]が広く普及している。SNMP ではモニタリング対象のサーバに SNMP エージェントが存在し、SNMP マネージャからの問い合わせに対して監視対象の状態を返すという手法を用いている。

ネットワーク経路のモニタリング手法には大きく分けてアクティブ型とパッシブ型の 2 種類が存在する。アクティブ型モニタリングではモニタリングモジュールが Probe パケットと呼ばれるテストパケットを送信し、帯域幅や遅延を測定する。最新の性能情報を取得できる反面、Probe パケットによるネットワーク負荷が発生するという問題点がある。代表的な手法には Pathchar [9], Bprobes[10], Cprobes[10]などがある。

パッシブ型モニタリングでは、経路を通る他のアプリケーションのトラフィックをモニタリングすることで情報の取得を行う[11]。そのためネットワーク負荷は一切発生しないが、即時性という点で問題がある。本研究では障害検知を迅速に行う必要から、アクティブ型モニタリングを利用するが、モニタリングの周期をアプリケーションの要求によって設定することで、ネットワーク負荷を低減する手法を提案する。

3. モバイルサービスプラットフォーム

本章では、2 章の要求条件に対して、オーバレイネットワーク型のサービスプラットフォームを示す。そして、アプリケーションの要求を満たし、且つネットワーク負荷を低減する、適応的モニタリングとモニタリング情報通知プロトコルの提案を行う。また、処理を切り替える際に問題となるサービスのインタフェースの違いを吸収する手法について述べる。

3.1 ネットワークアーキテクチャ

プラットフォームは複数のオーバレイノードから構成される(図 2)。サービスプロバイダによ

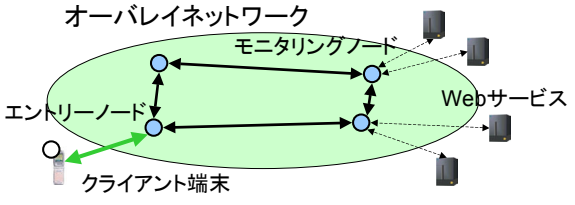

図 2 ネットワーク構成

って提供される各種の Web サービスは、最も近傍のオーバレイノードに登録されるものと仮定する。オーバレイノードは、管理対象のサービスノードや、サービスノードに至るまでのネットワークパスについて、障害や性能低下の検知のため分散的にアクティブ型のモニタリングを行う。モニタリングを行っているオーバレイノードをモニタリングノードと呼ぶ。一方、クライアント端末に最も近いオーバレイノードをエントリーノードと呼ぶ。エントリーノードはサービスのディスカバリや、障害発生時の代替サービスへの切り替えによる復帰処理を行う。以下、サービスディスカバリと、障害発生時の復帰処理に分けて、プラットフォームの機能について述べる。

サービスディスカバリを行う場合、まずクライアント端末上のアプリケーションが、必要とするサービスやネットワークの機能・QoS 要求についてのクエリを、エントリーノードに送信する。エントリーノードはクエリをオーバレイネットワーク上にフラッディングし、クエリを受信したモニタリングノードは自身の管理リストの中から要求を満たすサービスの情報を返す。エントリーノードは取得したサービスリストの中で最も性能の高いサービスを選択・使用すると同時に、それ以外のサービスを代替リストとして保持する。

障害発生時には、モニタリングノードが障害を検知し、エントリーノードに障害の発生を通知する。エントリーノードでは障害通知に対して、代替リストの中から次の候補を選択して切り替え、迅速なサービスの復旧を行う。

3.2 アプリケーション指向モニタリング

1. 適応的モニタリング

迅速に障害を検知しつつ、ネットワークへの負荷を最小限に抑えるために、本研究ではアプリケーション要求に基づいた適応的モニタリング手法を提案する。

障害を検知するためのモニタリング手法には Ping を用いた接続性、遅延、ロス率などの測定や SNMP エージェントをサーバ側に用意する手法、実際に SOAP の実行を行いサービスの応答性を測定するものまで、様々な手法が考えられる(表 1)が、それぞれに一長一短があるため、目的に応じて使い分ける必要がある。複数手法の併用なども考えられるが、この中でアクティブ型の Ping や SOAP 実行の手法について、以下に示す適応的モニタリング手法が適用可能である。

モニタリング手法	情報量	迅速性	ネットワーク負荷	サーバ負荷
Pingベース	△	○	○	○
SNMPエージェント	△	○	○	△
SOAP実行	○	○	×	×

表 1 モニタリング手法比較

クライアントアプリケーションは、サービスの障害からの回復時間についての要求を XML で記述し、エントリーノードに通知する。要求を受信したエントリーノードはモニタリングノードに要求を通知する。モニタリングノードは要求に応じてアクティブ型モニタリングを行う頻度を変更する。例えば、迅速な回復を要求された場合、モニタリングノードは対象となるリソースに対してモニタリングの頻度を高くして、迅速な障害検知を試みる。アプリケーションの要求に応じて、モニタリングの頻度の変更処理を行うことで、要求を満たしつつ、ネットワーク負荷を低減できる。

アプリケーション要求は、アプリケーションコード内に埋め込む方法や別ファイルで用意する方法やユーザが入力する方法などが考えられる。例えば、現在の i アプリでは ADF (Application Descriptor File)内に埋め込む手法がある。要求の記述方式は HQML [12]や QDL[13]など既存手法の使用を検討している。

モニタリングのタイムアウト時間(Tout)は、アプリケーションの回復要求時間(Treq)から、代替候補となるサービスまでの遅延(Tdelay)とサービス実行時間(Td)を引いた差に設定する(式 b)。このようにタイムアウト時間を設定することで、障害を検知した後に切り替え処理を行った場合でも、アプリケーション要求を満たしたサービス回復が可能となる。

$$Tout = Treq - (Tdelay + Td) \quad \cdots(b)$$

2. モニタリング情報通知プロトコル

モニタリングノードがモニタリングした情報をエントリーノードに通知する際のプロトコルについて、RDF を元にした動的リソース記述方式を用いてネットワーク負荷を低減する、通知プロトコルを提案する。

記述構造：動的リソース記述方式では、モニタリングによって得られた性能情報を動的変化要素として、リソースからの広告によって得られる静的要素と分けて構造化する。オーバレイネットワーク上に記述情報を送信する際には、

動的変化部分とヘッダ情報のみを差分通知することで、ネットワークへの負荷を軽減する。ヘッダ情報内には、リソースの URI が記述され、通知情報がどのリソースについての性能を表すかを示す。

正常・異常範囲記述：また静的要素の中に、サービス提供者から広告されたリソースの性能値に対する正常範囲・異常範囲情報の記述を含める。範囲情報にはリソースの性能値に対して、正常値の範囲や異常値の範囲が記述される。モニタリングノードは範囲情報と実測値を比較して性能低下や障害の発生を判断し、エントリーノードに対する障害通知を行う。この処理によって障害発生時にイベントドリブンに障害通知が行われるため、モニタリングによる検知のみの場合に比べて迅速な障害検知が可能になる。

代替ノード性能通知：さらに、代替候補となるサービスのモニタリング情報をエントリーノードに通知する際に一定の閾値を設けて、一定の水準を満たす代替サービスからのみモニタリング情報が通知される処理を行う。この結果、切り替えが行われることの少ない、性能値の低いサービスに関してはモニタリング情報が通知されず、ネットワーク負荷を軽減する効果が期待できる。

3.3 サービスインタフェースの変換

「同機能を提供するサービス」を代替サービスと定義しているが、プロセスの入出力インタフェースが同じであるとは限らない。入出力の型や単位の相違は頻繁に起こりうる。そのため、処理を切り替える際には切り替え先の入出力インタフェースと整合性を取るために要素の変換を行う必要がある。例えば、入力要素の温度の単位が摂氏と華氏で違っている場合や、文字列を 2 個入力するか、一つのクラスで入力するかといった相違は現在の Web サービスでも既に存在する。

一方、DAML-S[14]などのオントロジ記述言語では Web サービスやその入出力インタフェースについてオントロジをもたせ、他者との関係性を記述して、知的なサービスの検索や連携を行うことを目指している。現在、これを用いて入出力インタフェースの形式をオントロジ空間でマッピングすることで、相違の吸収を行うことを検討中である。

4. プロトタイプ実装

提案手法の有効性を検証するプロトタイプの実装を行った。より実環境に近い評価実験を行うため、テストベッドを構築した(図 3)。

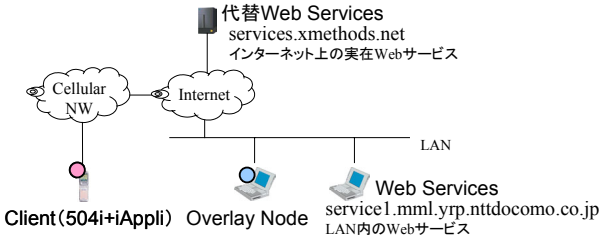

図 3 テストベッド構成

クライアント端末として PDC 端末 SO504i を、実際に使用する Web サービスとして、インターネット上のサービス(services.xmethods.net)と、ローカルサービス (service1.mml.yrp.nttdocomo.co.jp)を用意し、ローカルサービスと同じ LAN の内部にオーバレイノードを配置した。ローカルサービス側で擬似的に障害を発生させることで、障害検知及び障害からの復帰のシーケンスを確認した。

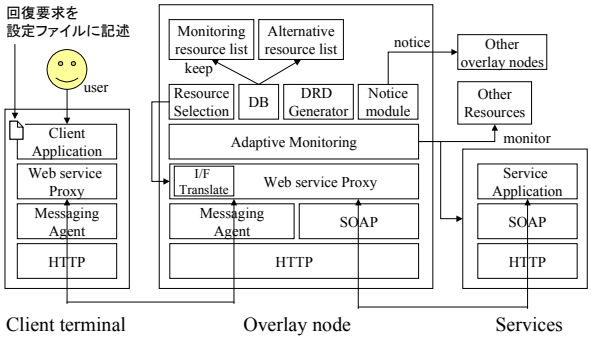

図 4 モジュール構成図

実装のモジュール構成図を図 4 に示す。クライアント端末上では i アプリによるユーザインタフェースと Web サービスプロキシ、株式情報を表示するアプリケーションが動作する。PDC 端末のリソースには制約があるため、軽量の Web サービスプロキシのみを実装し、オーバレイノード上の Web サービスプロキシと通信を行うことで機能を実現する。オーバレイノード上のプロキシは、クライアント端末からのデータを受け、SOAP で Web サービスの実行し、返された結果をクライアント端末に送信する機能を持つ。

オーバレイノード上に、障害時の適応機能として、適応的モニタリングモジュール、動的リソース記述作成・保持モジュール、代替リソース切り替えモジュールを実装した。適応的モニタリング

モジュールは、アプリケーションからの要求に応じてモニタリング頻度を変え、アクティブ型のモニタリングを行う。得られたモニタリング情報は動的リソース記述作成モジュールに渡され、動的リソース記述として保持される。障害時には、モニタリングモジュールが代替リソース切り替えモジュールに障害情報を通知し、切り替えモジュールはWebサービスプロキシに実行先のWebサービスを切り替えるよう指示する。切り替えに伴う入出力インタフェースの変換に関しては、今回の実験で用いるWebサービスのWSDLファイルを元にあらかじめ用意した変換モジュールを用いた。今後、サービスオントロジを用いてインタフェース変換処理を動的に行うことも検討する。

機能の実装は全てJavaで行い、オーバレイノード及びローカルサービスには東芝製Dynabook(CPU Pentium3 750MHz, Memory 128MB, OS Windows2000)を用いた。また、動的リソース記述作成・保持モジュールの作成にはHP社のRDFパーサであるJenaライブラリを用いた。

5. 評価

まず、テストベッドの動作確認を行った。クライアント端末からWebサービスプロキシを経由してローカル及び実サービスを実行し、結果をクライアント端末で表示できることを確認した。次に、サービスを実行する際にモニタリングの結果から応答性の高いサービスを選択して実行することを確認した。さらに、障害発生時に障害を検知し、サービスの実行先を切り替えることで復旧できることを確認した。

次に、実装したテストベッドを用いて、手法の有効性を検証するために 3 種類の評価実験を行った。まず、擬似的な障害を発生させ、障害を検知してから処理を代替サービスに切り替えてサービスが復帰するまでの時間を測定し、モニタリング頻度の変化と回復時間との関係を明らかにした。次に、モニタリングと動的リソース記述交換を行う際のネットワーク負荷を測定した。最後に、モニタリングノードでの動的リソース記述の作成処理がボトルネックになる可能性を考慮し、サービスノードの数とモニタリングノードの負荷について調べた。

5.1 障害検知から切り替えによるサービス復帰の時間測定

まず、モニタリングノードが障害を検知してから代替リソースへの切り替えによって実際にサービスが復旧するまでの時間を測定した。障害発生からサービス回復までの時間の内訳は 1 章の式(a)のようになる。

実験では適応的モニタリングによってモニタリングの頻度を変化させ、障害から復帰するまでの時間を測定した。また、テストベッドではモニタリングノードとエントリーノードを同一ノードとして実装したため、モニタリングノードからエントリーノードへの障害通知にかかる遅延が発生しない。今回の実験ではこの遅延を一律50msとした。

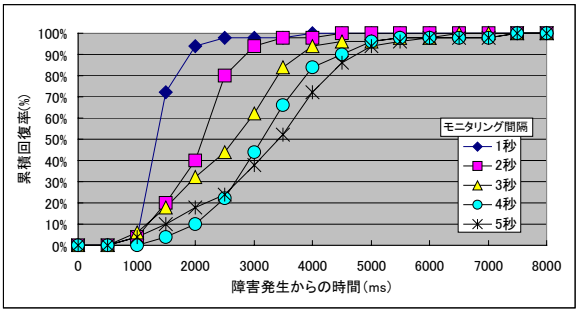

図 5 障害発生からサービス回復にかかる時

実験の結果を図5に示す。横軸は障害発生からサービス復帰までの時間を示し、縦軸はその時点での回復率を示す。

評価結果から、今回用いたサービスの場合はモニタリング頻度が2秒以下であれば、94%のサービスが3秒以内に回復することがわかった。音声通話などインタラクティブ性の高いサービスの場合には3秒の切断は問題となるが、数秒のバッファをクライアント側に保持すると思われるストリーミングサービスや、株価閲覧アプリケーションであれば、許容できる切断時間である。

他のサービスを使用する場合には 1 章の式(a)のうちサービス実行時間(Td)が変化し、それによって全体の回復時間が変化するため、単純に実験結果を流用することはできない。しかし、モニタリングノードが監視対象についてパッシブ型のモニタリングを行い、サービス実行時間(Td)が既知である場合は、全体の回復時間を計算して適切なモニタリング間隔を導くことができる。

5.2 モニタリングによるネットワーク負荷測定

次に、モニタリングを行うことでネットワークに与える負荷と、アプリケーションの回復時間に対する要求の達成率を測定した。モニタリングによる負荷は、1.モニタリングノードからモニタリング対象へのアクティブ型モニタリングによる

負荷と、2.障害発生時にモニタリングノードからエントリーノードに通知される動的リソース記述通知による負荷の 2 種類に分けられる。

実験では、1 秒以内の障害検知を要求するハードなリアルタイム系サービスと 5 秒以内の障害検知を要求するソフトなリアルタイム系サービスの 2 種類のサービスが混在している環境で、モニタリングノード-モニタリング対象間の適応的モニタリングによる負荷と、アプリケーション要求の達成率を調べた。

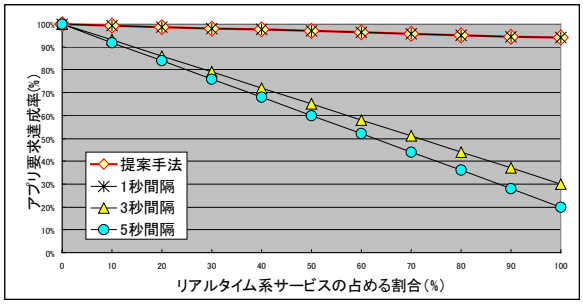

図 6 サービスの状態とサービス要求達成

図 6 にその結果を示す。横軸は、ハードリアルタイム系サービスとソフトリアルタイム系サービスの混在する割合を示し、縦軸はアプリケーション要求の達成率を示す。例えば、横軸が 80%の場合は、全サービス中 8 割がハードリアルタイム系サービスであり、残りがソフトリアルタイム系サービスであることを示している。

結果から、一定間隔のモニタリングでは、モニタリング間隔を大きくするにつれて要求達成率は落ち込むことが確認された。それに対して、適応的モニタリングはアプリケーションの要求を最大限に満たすよう、サービスごとにモニタリング間隔を調節するため、1 秒間隔での一定間隔モニタリングと同等の要求達成率を示した。

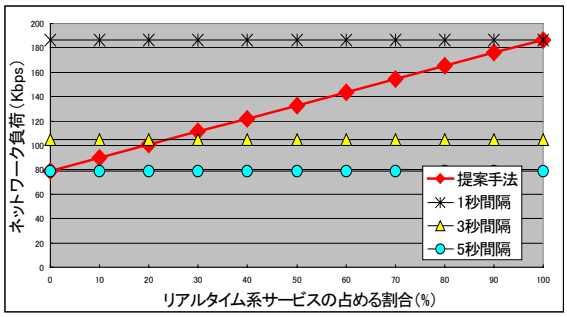

図7 サービスの状態とネットワーク負荷

図 7 に同条件でモニタリングがネットワークに与える負荷を測定した結果を示す。従来の一定間隔のモニタリングがアプリケーションの要求変化を考慮しないのに対して、適応的モニタリングはアプリケーション要求の変化に応じてネットワーク負荷を軽減する効果を示した。

以上のように、図 6、7 の結果から、提案手法である適応的モニタリングは、従来の一定間隔のモニタリングに対し、アプリケーションの要求を満たしながら、ネットワークに与える負荷を最小限に留めることが確認できた。

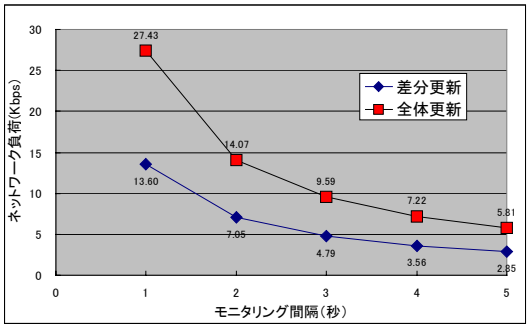

図 8 動的リソース記述の交換によるネットワーク負荷

次に、障害発生時にモニタリングノードからエントリーノードに通知される動的リソース記述のネットワーク負荷を測定した(図 8)。横軸は適応的モニタリングによって変化するモニタリング間隔を示し、縦軸はネットワーク負荷を示す。実験に用いた記述はモニタリングノードからサービスノードまでのネットワーク性能の記述情報で、全体のサイズが 2982 バイト、差分部分は 1302 バイトである(図 9)。実験結果から、モニタリング間隔の変化によらず、差分のみを更新する方式が、ネットワーク負荷を軽減することを確認した。

5.3 オーバレイノードの計算負荷測定

一つのモニタリングノードは多数のサービスを管理するため、モニタリングするリソースの数が増えるに従って負荷が増加し、ボトルネックになる可能性がある。そこで特に処理が重いと考えられる動的リソース記述の作成処理に着目し、モニタリング対象の数を変化させた場合の記述作成の負荷を測定した。

測定結果を図 10 に示す。横軸は一つのモニタリングノードが担当するリソースの数を示し、縦軸は記述の作成に要する時間を示す。実験の結果、サービスが増えるにしたがって、記述を作成するために要する時間が増加したが、2 回目以降のリソース記述の作成には、リソース数が 500 でも 1.8 ミリ秒しかかからなかった。これは、1 回目の記述作成時に記述のツリーがメモリ上に展開

```
<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:RDFNsId1='http://mml.yrp.nttdocomo.co.jp/mag3/drd#'>
  <rdf:Description rdf:about='stegoma-service2.dyndns.org'>
    <RDFNsId1:Service_Specification rdf:parseType='Resource'>
      <RDFNsId1:Dynamic_Elements rdf:parseType='Resource'>
        <RDFNsId1:Packet_Loss rdf:parseType='Resource'>
          <RDFNsId1:Last
            RDFNsId1:value='10'
            RDFNsId1:unit='%'/>
          <RDFNsId1:Average
            RDFNsId1:value='0'
            RDFNsId1:unit='%'/>
        </RDFNsId1:Packet_Loss>
        <RDFNsId1:Delay rdf:parseType='Resource'>
          <RDFNsId1:Last
            RDFNsId1:value='10'
            RDFNsId1:unit='ms'/>
          <RDFNsId1:Average
            RDFNsId1:value='100'
            RDFNsId1:unit='ms'/>
        </RDFNsId1:Delay>
      </RDFNsId1:Dynamic_Elements>
      <RDFNsId1:Static_Elements rdf:parseType='Resource'>
        <RDFNsId1:Overload rdf:parseType='Resource'>
          <RDFNsId1:Delay rdf:parseType='Resource'>
            <RDFNsId1:Range_type>upperbound
            </RDFNsId1:Range_type>
            <RDFNsId1:Range_value
              RDFNsId1:value='400'
              RDFNsId1:unit='ms'/>
          </RDFNsId1:Delay>
          <RDFNsId1:Packet_Loss rdf:parseType='Resource'>
            <RDFNsId1:Range_type>upperbound</RDFNsId1:Range_type>
            <RDFNsId1:Range_value
              RDFNsId1:value='20'
              RDFNsId1:unit='%'/>
          </RDFNsId1:Packet_Loss>
        </RDFNsId1:Overload>
        <RDFNsId1:Normal rdf:parseType='Resource'>
          <RDFNsId1:Delay rdf:parseType='Resource'>
            <RDFNsId1:Range_type>between</RDFNsId1:Range_type>
            <RDFNsId1:Range_value
              RDFNsId1:value='10'
              RDFNsId1:unit='ms'/>
            <RDFNsId1:Range_value
              RDFNsId1:value='250'
              RDFNsId1:unit='ms'/>
          </RDFNsId1:Delay>
          <RDFNsId1:Packet_Loss rdf:parseType='Resource'>
            <RDFNsId1:Range_type>lowerbound</RDFNsId1:Range_type>
            <RDFNsId1:Range_value
              RDFNsId1:value='20'
              RDFNsId1:unit='%'/>
          </RDFNsId1:Packet_Loss>
        </RDFNsId1:Normal>
      </RDFNsId1:Static_Elements>
    </RDFNsId1:Service_Specification>
  </rdf:Description>
</rdf:RDF>
```

動的記述差分

図 9 動的リソース記述例

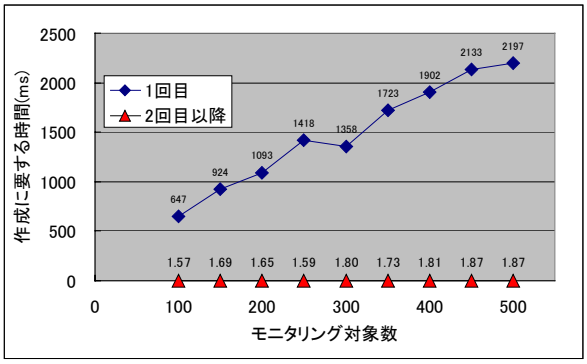

図 10 動的リソース記述作成に要する時間

されるためと考えられる。

現在の実装ではユーザがアプリケーションを立ち上げたと同時に該当するリソースに対してモニタリング処理が開始され、モニタリング結果を待ってからサービス実行が行われる。そのため、記述作成の時間がサービスの実行を遅らせるという問題がある。これに対して、メモリ消費の問題があるが、モニタリングノードが管理しているサービスの記述ツリーをあらかじめ作成しておくという対処が考えられる。

6. おわりに

本稿では、モバイル環境における安定した分散サービスの提供を課題とし、迅速かつネットワーク負荷を抑えた障害・性能低下の検知を可能とする適応的モニタリングと動的リソース記述方式を提案した。プロトタイプを用いて行った評価実験で、提案手法は従来手法と比較して、アプリケーションの要求を損なうことなく、ネットワーク負荷を最小限に抑制する効果があることを確認した。

今後、ステート保持の必要なサービスに対する切り替え手法や、インタフェースの相違を吸収するためのサービスオントロジを考慮したプラットフォームの構築などが課題となる。

参考文献

[1]David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris. ``Resilient Overlay Networks". Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[2]J. Touch. "Dynamic Internet Overlay Deployment and Management Using the X-Bone". Proc. ICNP 2000, Osaka Japan, pp. 59-68.

[3]白鳥則郎, 木下哲男, 菅原研次. やわらかいネットワーク. 情報処理学会誌, Vol.43, No.6, pp.639-644, 2002 年 6 月.

[4]John Kubiatowicz, et al. “Oceanstore: An Architecture for Global-Scale Persistent Storage”. Appears in Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.

[5]``Composite Capability / Preference Profiles (CC/PP): A user side framework for content negotiation". W3C Note. http://www.w3.org/ TR/NOTE-CCPP/. 27th, Jul 1999.

[6]``Web Services Description Language (WSDL) 1.1 W3C Note” http://www.w3.org/ TR/wsdl/. 15th March 2001.

[7]``Resource Description Framework (RDF) Model and Syntax Specification W3C Recommendation". http://www.w3.org/TR/ REC-rdf-syntax/. 22nd February 1999.

[8]D. Harrington, R. Presuhn, B. Wijnen: “An Architecture for Describing SNMP Management Frameworks”, RFC2271, 1998.

[9]”pathchar - A Tool to Infer Characteristics of Internet Paths”. ftp://ee.lbl.gov/pathchar, 1997.

[10]R. L. Carter and M. E. Crovella. “Measuring bottleneck-link speed in packet switched networks”. Technical Report BU-CS-96-006, Computer Science Department, Boston University, 1996.

[11]SESHAN, S., STEMM, M., AND KATZ, R. H. “SPAND: Shared Passive Network Performance Discovery”. In Proc. 1st USITS (Monterey, CA, December 1997), pp. 135-146.

[12] Xiaohui Gu, et al, An XML-based QoS Enabling Language for the Web, Journal of Visual Language and Computing, Academic Press, December, 2001.

[13] Joseph P.Loyall, et al. “QoS Aspect Languages and Their Runtime Integration”. Lecture Notes in Computer Science, 1511, May 1998.

[14]The DAML Services Coalition. “DAML-S: Web Service Description for the Semantic Web”. The First International Semantic Web Conference (ISWC), 2002