システムソフトウェアと
オペレーティング・システム 84−19
（2000．5．25）

端末およびサービス透過的な情報閲覧支援システムの構築

由良 淳一[1] 大越 匡[1] 中澤 仁[1] 徳田 英幸[1 2]
[1] 慶應義塾大学大学院 政策・メディア研究科 [2] 慶應義塾大学 環境情報学部

本研究では，端末およびサービス透過的な情報閲覧支援システムである，tranService システムを提案し，その設計と実装について述べる．本研究で提案する情報閲覧支援システムは，端末およびサービスの特性を考慮したメディア変換を行う．クライアントからの情報取得要求は，サービスに適するように変換される．これに対して，サービスからの情報は，ユーザ要求および端末の環境情報に適するように変換される．これにより，ユーザは新たなアプリケーションの導入を行うことなく情報を閲覧でき，サービス提供者はユーザや端末を考慮することなくさまざまな情報を提供できる．

Media Translation Based on QoS Requirement of Devices and Services

Jun'ichi Yura[1] Tadashi Okoshi[1] Jin Nakazawa[1] Hideyuki Tokuda[1 2]

[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University

This paper presents the design and implementation of information browsing support system transparent in devices and services, tranService. It translates the types of media in consideration of characteristic of devices and services. When a request for information is received from the client, it is altered to suit the format of the service. The information received from the service, in turn is altered to suit the user's request, and the information of the surrounding environment. Thus, the users can browse various information without any additional application, and service providers can provide information without consideration on users and devices.

1 はじめに

近年のコンピュータネットワークの普及は，インターネットの広がりにとどまらず，家電ネットワークやウェアラブルネットワーク [10] などの新しいネットワーク形態も誕生させ，それらの上で情報を提供するさまざまなサービスも出現している．Web サービスのような情報閲覧サービスは，文字や画像情報に限らず，音声や動画，アプレットなどのプログラムといったさまざまな情報を提供している．さらに，センサ群から継続的に提供される情報や，家電機器からのメディアデータなど，サービスが提供する情報の種類は，飛躍的に増えている．

また，コンピュータネットワークに接続可能な端末も多様化している．プロセッサや記憶媒体の小型化，省電力化により，ワークステーションやパーソナルコンピュータだけでなく携帯情報端末 (PDA: Personal Digital Assistants) や携帯電話，ウェアラブルコンピュータなどもコンピュータネットワークに接続可能になりつつある．

これらの多様な端末が，さまざまなサービスを受けるためにはいくつかの問題がある．

第 1 の問題として，端末間の性能差が挙げられる．例えば，カラーの動画を表示する場合は，ワークステーションやパーソナルコンピュータでは表示できるが，演算性能の低い PDA では表示できない．

第 2 の問題として，サービス間の特性差が挙げられる．端末からサービスが提供する情報を閲覧するには，その性質に応じたソフトウェアを導入する必要がある．例えば，Web を閲覧する場合とデータベースシステムからデータを取得する場合では，異なるアプリケーションが必要である．

端末間の性能の差は端末の性能に適した情報を得られない．また，サービス間の性質の差はサービスの利用を阻害し，情報を閲覧できないなどの問題を引き起こす．このため，端末やサービスの違いを吸収し，情報閲覧を支援する基盤ソフトウェアが必要となる．

1.1 研究の目的

本研究の目的は，端末の性能差およびサービスの性質差を吸収し，多種多様な情報をユーザが容易に閲覧できるシステムを構築することである．

本研究では，端末の性能差を吸収して情報閲覧を支援することを**端末透過的な情報閲覧支援**と呼び，サービスの特性差を吸収して情報閲覧を支援することを**サービス透過的な情報閲覧支援**と呼ぶ．図 1 に端末透過な情報閲覧の例を，図 2 にサービス透過な情報閲覧支援の例を示す．

端末透過とは，サービス提供者が端末の環境を考慮しなくても，ユーザが環境に適した情報を得られる性質である．図 1 の例では，あるサービスで提供されているカラー動画データが，電話には音声データに，PDA には白黒動画データに変換して送られる．

一方，サービス透過とは，ユーザがサービスの特

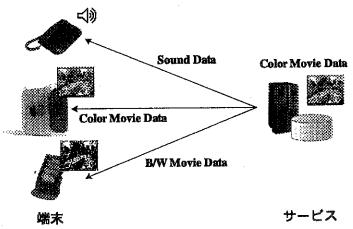

図 1: 端末透過

性を考慮することなく利用できる性質である．図2の例では，ニュースサービスやデータベースサービスなどが，Web クライアントで閲覧できるように変換される．

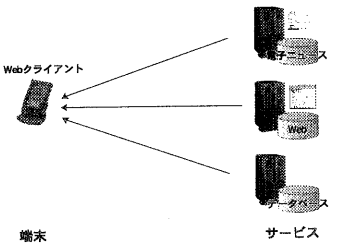

図 2: サービス透過

本研究では，クライアントからの要求データやプロトコルをサービスに適する形式に変換し，またサービスからのデータおよびプロトコルを端末に適する形式に変換する，tranService システムを開発した．本システムを利用することにより，ユーザはさまざまなサービスから，利用している端末に適したデータを取得できる．

本稿では，2章で情報閲覧支援の定義を行い，情報環境の問題点を述べる．3章で情報閲覧支援システムのモデル化とモデルの比較を述べる．4章では，tranService システムの概要および設計を述べ，5章で実装を述べる．6章では，定性的および定量的評価を述べる．最後に7章で，まとめと今後の課題を述べる．

2 情報閲覧支援

本研究における情報閲覧支援とは，あるサービス上で提供されている情報を，ユーザや端末に適するように変換を行うことである．本節では情報環境を定義し，情報環境の構成要素について説明する．また，情報閲覧支援を行う際に重要となる QoS プロファイルについても述べる．

2.1 情報環境

本研究で想定する情報環境とは，コンピュータネットワークに接続された，さまざまなサービスと多種多様な端末で構成された環境を指す．情報環境における端末とサービスについて以下に述べる．

2.1.1 端末

本研究では，ユーザの利用する，コンピュータネットワークに接続可能なハードウェアを端末と呼ぶ．端末は，プロセッサや記憶媒体の小型化，省電力化，低価格化により多様化しており，端末間の性能の差も大きくなっている．このような端末間の性能差として，以下のような点が挙げられる．

- **演算装置**
  中央演算装置や画像処理装置などの性能差．演算装置の性能が低いと，動画再生や画像の縮小，減色などの複雑な処理を行うのに時間を要する．
- **記憶装置**
  主記憶装置や二次記憶装置の容量差．記憶装置の容量が小さい端末は，マルチメディアデータの再生や保存には適さない．
- **通信環境**
  有線や無線の違いによる特性差，帯域の差による通信速度差．無線による通信環境は信頼が低く，接続が切断されることが多い．電話回線など低速な通信環境では大きなデータを取得するのに多くの時間を要する．
- **入出力デバイス**
  キーボードやマウス，ペン入力，音声入力などの入力手段の差．表示画面，音声出力などの出力手段の差．画面サイズや色数，ステレオ音声とモノラル音声などの性質の差がある．入力デバイスがないとデータ入力を行えず，出力デバイスがないとデータを表示，再生できない．

2.1.2 サービス

本研究では，コンピュータネットワーク上で情報を提供するソフトウェアをサービスと呼ぶ．コンピュータネットワークの普及に伴いサービスも多様化しており，サービス間の特性差は大きくなっている．サービス間の特性差の例を以下に挙げる．

- **データの種類や形式**
  サービスごとに扱うデータの種類や形式が異なる．データの種類としては，テキスト，画像，音声などが挙げられる．また，データの形式は，画像を例にとると，GIF 形式や JPEG 形式，PNG 形式などがある．
- **通信プロトコル**
  サービスごとに扱うアプリケーションプロトコルが異なる．例としては，Web サービスで利用される HTTP (Hyper Text Transfer Protocol)[3]，電子メールを送信する SMTP (Simple Mail Transfer Protocol)[7]，電子メールを受信する POP3 (Post Office Protocol version 3)[6]，動画データを配信する RTSP (Real Time Streaming Protocol)[8] などが挙げられる．

2.2 QoS プロファイル

情報環境において，サービスが提供する情報をユーザが閲覧できるためには，情報の品質が重要となる．例えば，あるサービスが動画像を提供していても，端末側でそれを再生できる機能が無ければ，ユーザはその情報を閲覧できない．情報品質の制約は，Quality of Service と呼ばれ，QoS と略される．

本研究では，QoS の要求内容を QoS プロファイルと呼ぶ．QoS プロファイルはデータプロファイルとプロトコルプロファイルの 2 つから構成される．データプロファイルはデータ自体の情報に関する情報であり，プロトコルプロファイルはデータの送受信方法に関する情報である．

データプロファイルは，データの種類および形式，データの性質により構成される．データの種類は MIME タイプ [2] の**type** 名であり，データの形式は**subtype** 名である．また，データの性質はデータの形式によって異なる．

プロトコルプロファイルは，プロトコルの形式およびプロトコルの性質により構成される．プロトコルには，HTTP や POP3 などのアプリケーションプロトコルの他にも，TCP/IP や IrDA などのネットワークプロトコルが含まれる．

3 情報閲覧支援システムモデル

情報閲覧支援システムとは，サーバで提供されている情報を端末やサービスの QoS プロファイルに適するように変換するシステムを指す．本節では，情報閲覧支援システムについてモデル化し，各モデルの比較検討を行う．

3.1 情報閲覧支援システムのモデル

情報閲覧支援システムは，情報を変換するコンピュータのネットワーク上の位置により，クライアント指向モデル，サーバ指向モデル，プロクシモデルの 3 つに分けられる．以下に情報閲覧支援システムの各モデルについて述べる．

- **クライアント指向モデル**
  クライアント指向モデルは，クライアント側 (ユーザ側) でデータやプロトコルの変換を行う情報閲覧支援のモデルである．例としては，クライアント側でさまざまなデータ形式を表示可能にする Web クライアントのプラグインモジュールが挙げられる．
- **サーバ指向モデル**
  サーバ指向モデルは，サービス側でデータやプロトコルの変換を行う情報閲覧支援のモデルである．例としては，ネットワーク帯域に応じた動画データを提供する，動画配信サーバが挙げられる．
- **プロクシモデル**
  プロクシモデルとは，サービスとクライアント間の通信の中継を行うプロクシサーバで，データやプロトコルの変換を行う情報閲覧支援のモデルである．例としては，小型端末用に画像を変換する Web プロクシサーバが挙げられる．

3.2 情報閲覧支援システムのモデル比較

クライアント指向モデルで端末透過性を実現するには，データ変換がクライアント側で行われるためデータは最適化されないまま端末まで送られる．処理能力の低い端末や帯域の狭いネットワークで利用する場合は，データ自体を変換できない場合がある．

サーバ指向モデルでサービス透過性を実現するには，各サービスごとにさまざまなクライアントに対する変換機能を持つことになる．これは，サーバ側の機能肥大化をまねき，非効率である．

また，端末透過性やサービス透過性を実現するために，サーバ指向モデルではサーバ上のソフトウェアの変更が，クライアント指向モデルではクライアントソフトウェアの変更が必要となる．これは，ユーザやサービス提供者にとって利用の阻害要因となる．

プロクシモデルでは，プロクシサーバ上のソフトウェアを変更する必要はあるが，サーバやクライアントのソフトウェアを変更する必要がない．また，変換機能を複数の端末やサービスから共有でき効率的である．

以上の理由により，本研究の対象とする情報閲覧支援システムには，プロクシモデルを採用した．

4 tranService システムの設計

本節では，端末透過性，サービス透過性を実現する tranService システムの設計について述べる．

4.1 ハードウェア構成

本システムのハードウェア構成は，インターネット等のコンピュータネットワーク接続を前提とし，PC やワークステーション，PDA などのさまざまな種類の端末と，Web や電子メール，データベースなどのさまざまなサービスを提供しているサーバの組み合わせを想定する．本システムのハードウェア構成を図 3 に示す．tranService システムは，端末とサービスの間にあるコンピュータ上で動作する．本システムは，端末からの要求をサービスに適するように変換し，サービスに要求を行う．また，サービスから送信されたデータを端末に適するように変換を行い，端末に送信する．

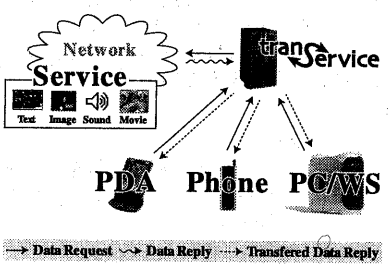

図 3: ハードウェア構成

4.2 ソフトウェア構成

本システムは，大きく分けて端末モジュール部，変換モジュール部，サービスモジュール部の 3 つのサブシステムで構成される．図 4 に各サブシステムの関係を示す．

4.2.1 モジュール部

本システムでは，モジュール群とそれらを管理するコントローラをモジュール部と呼ぶ．コントローラは，モジュールの追加や削除，受け取ったデータソースのモジュールへの振り分けなどを行う．各モジュール部の説明を以下に示す．

端末モジュール部

端末モジュール部は，クライアントとの通信を行うモジュール群およびそれらを統括するコント

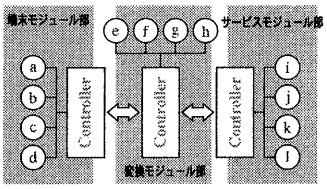

図 4: ソフトウェア構成

ローラから成る．モジュールは，クライアントが利用するプロトコルごとに用意される．クライアントから送信された要求および作成した QoS プロファイルを変換モジュール部に渡す．また，変換モジュール部から受け取ったデータソースをクライアントに送信する．

サービスモジュール部

サービスモジュール部は，サービスとの通信を行うモジュール群およびそれらを統括するコントローラから成る．モジュールは，サービスが利用するプロトコルごとに用意される．変換モジュール部から受け取った要求をサービスに送信する．また，サービスから受け取ったデータソースおよび作成した QoS プロファイルを変換モジュール部に渡す．

変換モジュール部

変換モジュール部は，データやプロトコルを変換するモジュール群およびそれらを統括するコントローラから成る．端末モジュール部から受け取った要求をサービスに適する形式で変換を行いサービスモジュール部に渡す．また，サービスモジュール部から受け取ったデータをクライアントに適するように変換を行い端末モジュール部に渡す．

4.2.2 データソース

モジュール間でやり取りされる情報をデータソースと呼ぶ．データソースは，コンテンツ，コンテンツプロファイル，端末 QoS プロファイル，サービス QoS プロファイルで構成される．

コンテンツ

コンテンツは，クライアントアプリケーションから取得した要求データ，またはサービスから取得した応答データである．コンテンツは端末モジュール部またはサービスモジュール部で取得し，変換モジュール部で変換される．

コンテンツプロファイル

コンテンツプロファイルは，コンテンツの大きさや種類などのコンテンツに関する付加情報から成る．コンテンツプロファイルは端末モジュール部またはサービスモジュール部で作成され，変換モジュール部でコンテンツを変換する際に利用される．

端末 QoS プロファイル

端末 QoS プロファイルは，ユーザ要求や端末の環境情報から成る．端末 QoS プロファイルは端末モジュール部で作成され，端末透過性を実現するために利用される．サービスから取得した応答データは，端末 QoS プロファイルに適するように変換される．クライアントに返されるデータのコンテンツプロファイルは，端末 QoS プロファイルと一致する．

サービス QoS プロファイル

サービス QoS プロファイルはサービスからの要求やサーバの環境情報から成る．サービス QoS プロファイルはサービスモジュール部で作成され，サービス透過性を実現するために利用される．クライアントアプリケーションから取得した要求は，サービス QoS プロファイルに適するように変換される．サービスに送られる要求のコンテンツプロファイルは，サービス QoS プロファイルと一致する．

4.3 動作手順

ここでは，tranService システムのモジュール群の動作手順を述べる．図 5 にクライアントとサービス間の動作手順を示す．

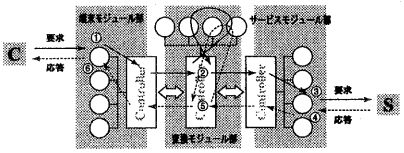

図 5: 動作手順

クライアントとサービス間の動作手順は以下の順に行う．

1. 端末モジュールがクライアントから送信された要求データからデータソースを作成する．
2. 変換モジュールコントローラがサービス QoS プロファイルから適切な変換モジュールを選択し実行する．各変換モジュールがデータソースを変換する．
3. サービスモジュールがデータソースを解析し，サーバに要求を送信する．
4. サービスモジュールがサーバから送信された応答データからデータソースを作成する．
5. 端末モジュールコントローラが端末 QoS プロファイルから適切な変換モジュールを選択し実行する．各変換モジュールがデータソースを変換する．
6. 端末モジュールがデータソースを解析し，クライアントに応答を送信する．

4.4 QoS プロファイルの設計

tranService システムでは，ユーザやサービスの要求，ハードウェアに関する情報である QoS プロファイルを用いてメディアデータの変換を行う．

QoS プロファイルは木構造のデータであり，データプロファイルとプロトコルプロファイルから構成される．図 6 に QoS プロファイルの例を示す．

QoS プロファイルの値の設定は各モジュールに任されている．例えば，HTTP 端末モジュールは，`Accept` タグや`User-Agent` タグの内容から QoS プロファイルの値を設定する．しかし，プロトコルでユーザ要求を指定することが出来ない場合やさらに細かいユーザ要求を行いたい場合は，QoS プロファ

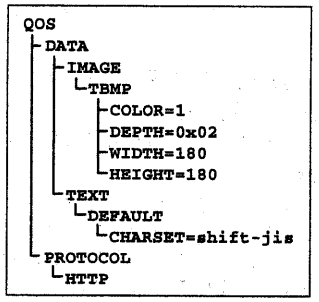

図 6: QoS プロファイルの例

イルをクライアントが明示的に送信する必要がある．本システムでは，QoS プロファイルをシステム外に送信したり，システム外より受信するために，XML (eXtentible Markup Language)[9] を利用している．

4.5 変換経路の決定

例として，サービスから**http** プロトコルを利用して送られてきた画像ファイルをテキストデータに変換し，クライアントに**pop3** プロトコルで送る場合を想定する．

変換経路決定の前提

クライアントが利用するプロトコルは**pop3** である．また，クライアントで利用可能なコンテンツ形式は**text/plain** で，文字コードは**iso-2022-jp** である．サーバが利用するプロトコルは**http** である．また，クライアントの要求によりサーバより送られるコンテンツは画像ファイル (コンテンツ形式は **image/gif**) である．

本システムの変換モジュールコントローラには，表 1 に示す 4 つの変換モジュール (変換モジュール (a)〜(d)) がインストールされている．また，サービスから受け取ったデータソースを端末に適切な形式に変換を行う．

表 1: 変換モジュール

	変換形式	入力形式	出力形式
a	データ種類変換	image/gif	text/html
b	データ形式変換	text/html	text/plain
c	データ品質変換	text/plain shift-jis	text/plain iso-2022-jp
d	プロトコル変換	http	pop3

変換経路決定手順

以下に，上記の仮定からなされる変換経路決定の手順について述べる．

変換モジュールコントローラに渡された時のコンテンツ QoS プロファイルは，図 7 の 1 である．端末 QoS プロファイルは図 7 の 5 であるので，変換終了後にコンテンツプロファイルが端末 QoS プロファイルと同じになるように変換経路を決定する．

i. 端末 QoS プロファイル (図 7 の 5) が出力である変換モジュールを探し，変換モジュール (d) が決定される．本システムでは，入力形式と出力形式の対は一意であるので，複数の変換モジュールが選択されることはない．
ii. 変換モジュール (d) の入力である QoS プロファイル (図 7 の 4) が出力である変換モジュールを探し，変換モジュール (c) が決定される．
iii. 同じようにして，変換モジュール (b)，変換モジュール (a) が決定される．
iv. 最終的には，変換モジュール (a)→ 変換モジュール (b)→ 変換モジュール (c)→ 変換モジュール (d) の順番に変換を行うことで，コンテンツが端末 QoS プロファイルに適するように変換される．

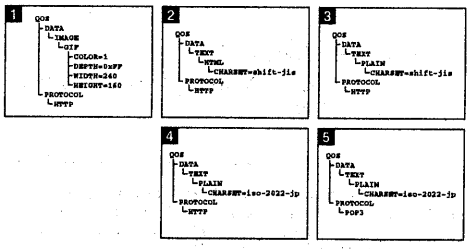

図 7: 変換中の QoS プロファイルの変化

5 tranService システムの実装

本節では，tranService プロトタイプシステムの実装について詳述する．

5.1 実装方針

tranService システムの実装は，プラットフォーム独立性と機能拡張性の面から Java 言語 [5] を選択した．

Java 言語で実装したプログラムは，JVM(Java Virtual Machine) が動作するコンピュータであればプログラムを再コンパイルすることなく実行できる．Java 言語で実装されたモジュールは他の JVM が動作する計算機でも利用でき，再利用性の面からも有用である．

JVM ではクラスローダと呼ばれるクラス読み込み機構によりクラスファイルの読み込みが行われる．クラスローダは JVM に組み込まれたもの以外に，ユーザ定義のものを利用できる．ユーザ定義のクラスローダを利用することで，ファイルやネットワークを介してモジュールを読み込むことができ，機能拡張性を実現する．

5.2 実装の概要

tranService プロトタイプシステムは，Java 1.2.1 で実現され，約 5000 行から成っている．本システムの実装の概要を図 8 に示す．

モジュールはサーバモジュールとワーカモジュールに分かれており，サーバモジュールがワーカモジュールを管理している．サーバモジュールはデータを受け取ると，ワーカモジュールを新たに作成し，データソースを渡す．モジュールコントローラはサーバモジュールの管理を行うほか，受信したデータソースをサーバモジュールに渡す．

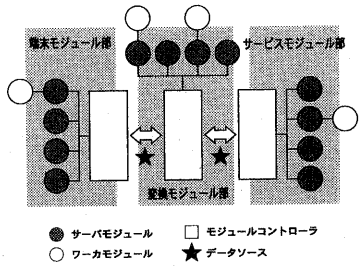

図 8: 実装の概要

5.3 モジュールの記述方法

図 9 に**text/html** から**text/plain** に変換するモジュールの例を示す.

```
import jp.ac.keio.sfc.ht.tranService.*;
public class SampleTranslationModule
             extends TranslationWorkerModule {
  public SampleTranslationModule() {
    // 変換形式はデータ変換.
    setTranslationType(TranslationServerModule
                       .TRANSLATION_TYPE_DATA);
    // 入力形式は"text/html"
    setInputType(new MimeType("text", "html")
                                    .toString());
    // 出力形式は"text/plain"
    setOutputType(new MimeType("text", "plain")
                                    .toString());
  }
  public void execute() {
    try {
      // データソース待ち受け
      Data data = getNextData();
      // ここで変換処理を行う.
      ... 省略 ...
      // データソース送信
      sendData(data);
    }
    catch(InterruptedException ie) { ... }
  }
}
```

図 9: 変換モジュールの例

変換モジュールでは, コンストラクタで変換形式および入力形式, 出力形式を設定する. データ変換モジュールでは, 変換形式に**TranslationServerModule.TRANSLATION_TYPE_DATA** を指定し, 入力形式および出力形式は MIME タイプで指定する. プロトコル変換モジュールでは, 変換形式に**TranslationServerModule.TRANSLATION_TYPE_PROTOCOL** を指定し, 入力形式および出力形式をプロトコルを文字列で指定する.

変換モジュールコントローラがデータソースを受け取り, **text/html** から**text/plain** に変換する必要がある場合, サーバモジュールを経由して, このクラスの**execute** メソッドを呼ぶ. データソースを受け取るには, 親クラスの**getNextData** メソッドを呼び, データソースを取得する.

変換モジュールは, データソース内のコンテンツの変換処理を行い, 親クラスの**sendData** メソッドを呼びだしてデータソースをサーバモジュールに渡す.

6 tranService システムの評価

本節では, tranService プロトタイプシステムの評価を, 定量的評価と定性的評価に分けて行う.

6.1 定量的評価

tranService システムを用いて構築された, 情報閲覧支援プロクシサーバの処理速度について定量的評価を行った. 本測定により, 本システム内部のオーバヘッドおよび, 変換モジュール数の違いによる性能面への影響などに関する基本的性能特性を把握する. 測定結果およびその考察を今後の実装最適化の一助とすることが本測定の目的である.

6.1.1 測定環境

定量的評価のための測定環境として, 表 2 に示す計算機を用いた. 測定には, ネットワークを利用せず, 全て同一計算機で行った. これは, 本測定の目的が, システム内部のオーバヘッドの測定であり, ネットワークを利用した場合の外部的影響を排除するためである.

表 2: 測定環境

CPU	UltraSPARC-II 248MHz × 2
主記憶	512MB
OS	Solaris7

6.1.2 測定方法

測定用モジュールには, HTTP プロトコルを解析する端末モジュールおよびサービスモジュールを利用した. また, HTTP サーバと HTTP クライアントを作成した. HTTP クライアントは本システムを経由して HTTP サーバから 1000 バイトのデータを取得する. クライアントからは, 図 10 の QoS プロファイルを本システムに送る. この QoS プロファイルでは, 全てのデータを**text/plain** 形式で要求する. 本システムでは, その形式に適するように変換を行う.

測定は, 本システムのソースコード内各所にタイムスタンプを**System.currentTimeMillis** メソッドを用い, ミリ秒の粒度で各処理単位ごとの所要時間を計算した. タイムスタンプを記録した場所を, 図 11 に示す. また, 表 3 に各所の説明を示す.

```
<QOS>
<DATA><TEXT><PLAIN></PLAIN></TEXT></DATA>
<PROTOCOL><HTTP></HTTP></PROTOCOL>
</QOS>
```

図 10: 測定用の QoS プロファイル

以上の方法により, 以下の 2 点について測定を行った.

- **測定 1: 全体的な処理時間の測定**
  変換モジュールを追加せずに, 本システムの全体

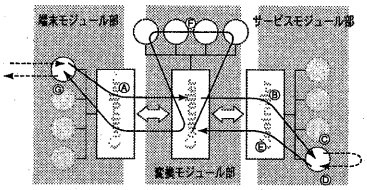

図 11: 測定方法

表 3: 測定個所の説明

	説明
A	端末モジュールがクライアントからデータを受信し，データソースを作成する．端末モジュールコントローラを経由して変換モジュール部にデータソースを送信する．
B	変換モジュール部は何の変換も行わず，サービスモジュールコントローラ経由でサービスモジュールに送信する．
C	サービスモジュールがサービスにデータを要求する．
D	サービスモジュールがサービスからデータを受信する．
E	サービスモジュールがデータソースを作り直し，サービスモジュールコントローラ経由で端末モジュール部に送信する．
F	変換モジュールがあれば変換モジュールにデータソースを送信する．全ての変換モジュールに送信すると，端末モジュールコントローラ経由で端末モジュールに送信する．
G	端末モジュールがクライアントにデータを送信する．

的な処理時間の測定を行う．本測定により，システム全体のオーバヘッドを計測する．

- **測定 2: モジュール数変化による処理時間の測定**
  変換処理を行わないモジュールを 0 個，10 個，25 個，50 個追加し，連続して変換するように入出力タイプを設定した．本測定により，モジュール数増加による変換経路決定の処理時間の計測を行う．

6.1.3 測定結果

本節では，前節の方法に基づいて行った測定の結果および考察を述べる．

測定 1: 全体的な処理時間の測定

本システムの各個所の所要時間を，表 4 に示す．本測定では，変換モジュールは追加しない．また，測定結果をグラフ化したものを，図 12 に示す．

表 4: 測定 1 の結果

測定個所	A	B	C	D	E	F	G
測定値	14.5	1.8	5.5	30.9	0.7	2.3	1.6
割合	25.3	3.1	9.6	54.0	1.2	3.9	2.9

(A) で処理時間比率が大きいのは，クライアントから送信されたデータの読み込みと，読み込まれたデータを解析部分に処理時間を要するのが原因である．特に，クライアントから送信された XML 形式の QoS プロファイルを，QosProfile クラスに変換している部分に処理時間を要した．また，(D) で処理時間が大きいのは，サービスからのデータの読み込みに時間を要するのが原因である．

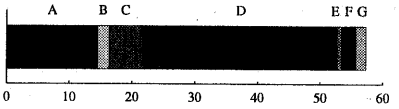

図 12: 測定 1 のグラフ

この結果から，上の 2 点以外の処理時間比率は小さいものであるため，本システムのパフォーマンスの向上のためには，ネットワークからのデータの読み書きおよび QoS プロファイルの変換処理の時間短縮が必要であることが分かった．

測定 2: モジュール数変化による処理時間の測定

変換モジュール数を，0 個，10 個，25 個，50 個と変化させた場合の，全体の所要時間および変換モジュール部の所要時間，変換経路決定の処理時間を，表 5 に示す．また，測定結果をグラフ化したものを，図 13 に示す．

表 5: 測定 2 の結果

モジュール数	0	10	25	50
全体	57.23	78.6	115.23	186.93
モジュール処理	2.25	18.94	49.42	98.65
変換経路決定	0	6.7	13.12	34.95

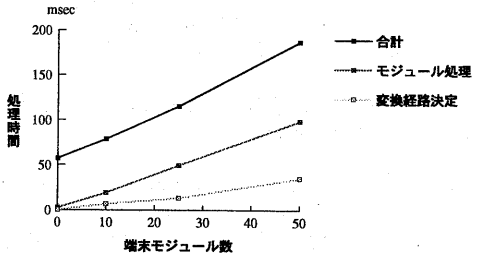

図 13: 測定 2 のグラフ

この結果を線形近似によって数式化したものを式 1 から式 3 に示す．

$$T_{sum} = 2.6Num + 54 \quad (1)$$

$$T_{exec} = 1.9Num + 1 \quad (2)$$

$$T_{calc} = 0.7Num - 1 \quad (3)$$

式 1 は，本システム全体の処理時間 (T_{sum}) を変換モジュール数 (Num) で表したものである．また，式 2 はモジュール処理時間 (T_{exec}) を，式 3 は変換経路決定処理時間 (T_{calc}) を表したものである．

式 3 から，変換モジュールが 1 つ増えると，モジュール検索時間が 0.7msec 増えることが分かる．また，式 2 から，モジュール 1 つの増加でモジュー

ルの処理時間が1.7msec増えていることが分かるが，この値は変換処理内容によって変動する．

モジュール数が増加した場合の処理時間の変動は2つの要因からなる．一つはモジュールの起動，変換などのモジュール自体の処理であり，もう一つはモジュールを選択するための検索時間である．モジュール自体の処理は，モジュールの処理内容によって処理時間が変わるため，処理時間短縮のためには，処理自体を最適化する必要がある．また変換経路選択のための検索時間は，検索アルゴリズムを最適化したり，キャッシュ機構を用意し変換経路を保存しておくことで，短縮できる．

6.2 定性的評価

本節では，類似システムとtranServiceとの機能評価を行う．評価項目および評価結果を表6に示す．

表 6: 定性的評価

サービス透過性

	W	M	T	D	T2
動的適応性	×	×	×	×	○
機能拡張性	×	×	×	○	○
変換連続性	×	×	×	×	△
プラットフォーム独立性	○	○	×	×	○

端末透過性

	W	M	T	D	T2
動的適応性	△	○	○	×	○
機能拡張性	○	○	○	○	○
変換連続性	△	×	×	×	○
プラットフォーム独立性	○	○	×	×	○

W:WBI[1], M:KMSF-MCAP[12], T:TranSend[4], D:Delegate[11], T2:tranService

○:実現可能，△：一部実現可能，×:実現不可能

以下に，サービス透過性，端末透過性の両面から本システムの機能評価を行う．

サービス透過性

本システムは，サービスのQoSプロファイルを考慮してデータやプロトコル変換を行うことで，サービス透過性を実現している．プロトコル変換を行う点ではDeleGateも同じであるが，DeleGateではサービスの要求を反映するような機構は存在せず，動的適応性という点で本システムの方が優れている．また，本システムでは変換連続性は実装されていないが，今後実装予定である．

端末透過性

本システムは，端末のQoSプロファイルを考慮してデータやプロトコル変換を行うことで，端末透過性を実現している．多くの類似システムでも端末透過性の機能は実装されている．WBIではモジュールを追加できることから機能拡張性があるが，その他の機能は提供されておらず，プログラマが独自に行う必要がある．TranSendやKMSF-MCAPではユーザや端末の要求をデータ変換に反映させている点で，本システムと同様に動的適応性や機能拡張性はあるが，変換連続性はない．

本システムでは，Java言語を利用しているため，プラットフォーム独立性や動的適応性がある．

以上のことから，本システムが他の類似システムに比べて機能的に優れていることが分かった．

7 おわりに

本研究では，端末やサービスのQoSを考慮した情報閲覧支援システムである，tranServiceシステムを提案し，設計と実装，評価を行った．本システムを利用することにより，ユーザがサービスの特性を考慮することなく自分の端末に適した形式で情報を取得，閲覧することができる．また，モジュールのプログラマも，本システムの柔軟性および動的拡張性により，容易に本システムを利用できる．

今後の課題としては，連続メディアデータへの対応やQoSプロファイルの互換性向上が挙げられる．既存のデータに対応することによって，本システムの利用範囲が広がることが予測される．また，モジュールをネットワークを介して利用することにより，負荷分散や耐故障性を実現できると思われる．

参考文献

[1] Rob Barrett and Paul P. Maglio. Intermediaries: New places for producing and manipulating web content. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[2] N. Borenstein and N. Freed. *MIME (Multipurpose Internet Mail Extensions)*, 1992. RFC 1341.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frysty, and T. Berners-Lee. *Hypertext Transfer Protocol - HTTP/1.1*, 1997. RFC 2068.

[4] Armando Fox, Steven D. Gribble, Yatin Chawathe, and Eric A. Brewer. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. In *IEEE Personal Communications*, September 1998.

[5] J.Gosling, B.Joy, and G.Steele. *The Java Language Specification*. Addison Wesley, Reading, Massachusetts, 1999.

[6] J. Myers and M. Rose. *Post Office Protocol - Version 3*, 1996. RFC 1939.

[7] Jonathan B Postel. *Simple Mail Transfer Protocol*, 1982. RFC 821.

[8] H. Schulzrinne, A. Rao, and R. Lanphier. *Real Time Streaming Protocol (RTSP)*, 1998. RFC 2326.

[9] W3C. *Extentible Markup Language*, 1997. http://www.w3.org/.

[10] 岩本健嗣, 西尾信彦, 徳田英幸. ウェアラブルコンピュータにおけるミッション機構. 情報処理学会コンピュータシステム・シンポジウム論文集, pp. 41–48, 1999.

[11] 佐藤豊. プロトコル中継システム DeleGate の開発. 電総研研究速報, 1994. TR-94-17.

[12] 牧野, 由良, 原嶋, 金平, 大越, 中澤, 徳田. Keio Media Space Family - MCAP システム. 情報処理学会コンピュータシステム・シンポジウム論文集, pp. 83–88, 1997.