卒業論文　1999年度(平成11年度)

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

指導教員
慶應義塾大学環境情報学部
徳田　英幸
村井　純
楠本　博之
中村　修

慶應義塾大学環境情報学部
由良 淳一

卒業論文要旨　　1999年度(平成11年度)

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

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．

近年のコンピュータネットワークの普及により，そのネットワーク上で情報を提供するWebやデータベースシステム，センサ群など，さまざまなサービスが存在する．さらに，コンピュータネットワークに接続可能な端末も，パーソナルコンピュータやワークステーション，携帯情報端末，ウェアラブルコンピュータなど多様化している．しかし，サービス間の性質差や端末間の性能差により，さまざまな端末から多様なサービスを閲覧することは困難である．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システムが必要である．

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．

本システムでは各構成要素がモジュール化されているため，変換処理の追加や拡張できることも特徴である．新しいデータ形式やプロトコルが登場しても，モジュールを追加することにより対応できる．また，各モジュールを組み合わせることにより，さらに多くの変換処理を柔軟に行える．

本論文では，まず情報閲覧支援の必要性を述べ，既存の情報閲覧支援システムの問題点を指摘する．次に，この問題点を解決する，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の設計，実装について述べる．最後に定量的および定性的評価をもとに，本システムの有用性を実証する．

慶應義塾大学　環境情報学部
由良 淳一

Abstract of Bachelor's Thesis

Media Translation Based on QoS Requirement of Devices and Services

This thesis presents the design and implementation of information browsing support system transparent in devices and services, tranService.

With the spread of computer networks, many types of services that supply information on the network such as the World Wide Web, database system, and sensors exist. A variety of network reachable device alternatives are becoming available such as personal computers, workstations, PDAs, and wearable computers. However, due to the limitations of capabilities and performance between different services and devices, if is difficult to browse the various services from different devices. For this reason, an information browsing support system that absorbs the differences between the two is necessary.

The information browsing support system proposed in this thesis 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. With this, the users can browse various information without having to install any new application, and service providers can provide information without having to considering users and devices.

Because the components of tranService are divided into modules, it is possible to add or extend translation transactions dynamically. Even if new data structure or protocol becomes available, there is only the need to add the new modules. By combining several modules, it is possible to execute more translation transactions with flexibility.

In this paper, first the necessity of information browsing support, and problems concerning existing information browsing support system is discussed. Furthermore, the design and implementation of tranService is described, and finally, possibilities of tranService are discussed based on the result of evaluation.

Jun'ichi Yura

Faculty of Environmental Information
KEIO University

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

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などの新しいネットワーク形態も誕生させ，そのネットワーク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提供サービスは，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．また，センサ群から継続的に提供される情報や，家電機器からの制御データなど，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(PC: Personal Computer)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．

このような多様な端末が，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，サービス間に特性の差が挙げられる．端末からサービスを閲覧するには，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．

第2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，性能の低いPDAでは表示できない．

このように，サービス間の性質の差はサービスを閲覧できないなどの問題を引き起こす．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．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ような基盤ソフトウェアが必要となる．

1.2　本研究の目的

本研究の目的は，端末の性能の差およびサービスの性質の差を吸収し，多種多様な情報を閲覧できるシステムを構築することである．

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よう

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ように変換する，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．

本システムによって実現される情報閲覧支援システムの概念図を図1.1に示す．端末から通信プロトコルAによる要求Aが本システムに送られる．その後，ユーザの要求するサービスに適するように，要求Aは要求Bに，プロトコルAはプロトコルBに変換する．変換を終えると，サービスにプロトコルBによって要求Bを送信する．サービスから送られたデータも，同様に本システムで変換して端末に送信する．

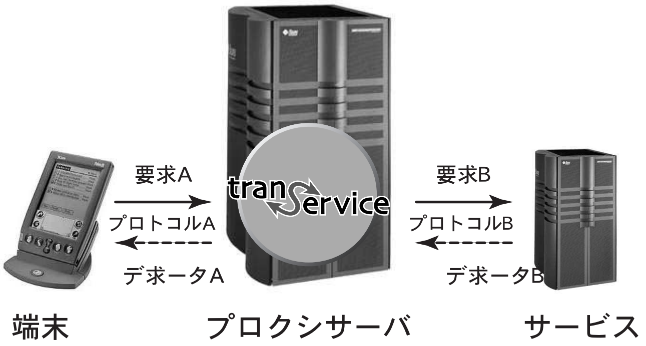


図 1.1: tranService のシステム概念図

1.3　本論文の構成

本論文では，第2章で情報環境と情報閲覧支援の定義を行った上で，情報閲覧支援の必要性について述べる．

第3章では，既存の情報閲覧支援システムをクライアント指向，サーバ指向，プロクシの3つのモデルに分類し，比較検討を行う．またプロクシモデルの情報閲覧支援システムについて解説し，既存のプロクシモデルの情報閲覧支援システムについて問題点を明らかにする．

第4章では，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システムの設計方針，システム構成，動作手順，QoSプロファイルと各モジュール部の設計について述べる．

第5章では，tranServiceプロトタイプシステムの実装環境，QoSプロファイルと各モジュール部の実装の詳細を述べる．また，本プロトタイプシステムを用いた情報閲覧支援のためのモジュールの実装方法についても述べる．第6章では，実装したプロトタイプシステムの測定と評価結果の考察，類似研究との機能比較を行う．

第7章では，本研究によって得られた結論と，今後の課題について述べる．

第2章　情報閲覧支援

本章では，情報閲覧支援について現状を述べた上でその必要性を説明する．

まず，本論文が想定する情報環境について定義し，情報環境の構成要素について説明を行う．最後に，情報閲覧支援の定義を行い，情報閲覧支援の際に重要となるQoSプロファイルについて説明する．また，情報閲覧支援の種類，必要性についても述べる．

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が行えない．
- 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保存，再生には適さない．
- 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．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がかかる．
- 入出力デバイス
  入力は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．出力は，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，モノラル音声などの性質の差もある．入力デバイスがないとデータの要求を行えないし，出力デバイスがないとデータを表示，再生できない．

表 2.1 に端末間の性能差についてまとめたものを示す．

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，画像を例にすると，GIF 形式や JPEG 形式，PNG 形式などがある．

表 2.1: 端末の性能差

	PC	PDA	携帯電話
演算装置	PentiumIII 550MHz	MC68328 16MHz	32bitRISC CPU 10.7MHz
主記憶装置	512MB	4MB	1024KB
通信環境	有線 100Mbps	有線 28kbps	無線 9600bps
表示デバイス	1280×1024 24bit カラー	160×160 2bit 白黒	100×100 2bit 白黒
音声出力	ステレオ	モノラル	モノラル

PC: IBM 社製 IntelliStation[8]，PDA: IBM 社製 WorkPad[17]，
携帯電話: 富士通製 F501i[3]

- 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 サービスではHTTP (Hyper Text Transfer Protocol)[4]，メールを送信するSMTP (Simple Mail Transfer Protocol)[13]，メールを受信する POP3 (Post Office Protocol version 3)[12]，動画データを配信するRTSP (Real Time Streaming Protocol)[15] などが挙げられる．

表 2.2 にサービス間の特性差についてまとめたものを示す．

表 2.2: サービスの特性差

	Web	電子メール	動画配信
通信プロトコル	HTTP	SMTP, POP3	RTSP
データの種類	テキスト，画像，音声，動画	テキスト	動画

2.2 情報閲覧支援

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．

この節では，変換を行う際に重要となる QoS プロファイルについて説明し，情報閲覧支援の種類と必要性について述べる．

2.2.1 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プロファイルは要求内容の性質や要求元の違いから分類できる．ここでは，各分類について説明する．

QoSプロファイルの構成

本研究では，QoSプロファイルはQoSの要求内容の性質からデータプロファイルとプロトコルプロファイルの2つに分類した．データプロファイルはデータ自体の情報に関するQoSプロファイルであり，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QoSプロファイルである．以下にそれぞれの説明を行う．

- データプロファイル
  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．表2.3にデータプロファイルの例を示す．

表2.3: データプロファイルの例

データの種類	データの形式の例	データの性質の例
`image`	`gif`，`jpeg`，`png`	大きさ，色深度
`audio`	`aiff`，`wav`，`au`	ステレオ/モノラル，サンプリングレート
`text`	`plain`，`html`，`xml`	文字コード

- プロトコルプロファイル
  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．

QoSプロファイルの種類

QoSプロファイルのもう一つの分類は，QoSの要求元の違いからの分類である．QoSの要求元としては，ユーザ，アプリケーション，デバイスの3つに分類できる．それぞれをユーザQoSプロファイル，アプリケーションQoSプロファイル，デバイスQoS

プロファイルと呼ぶ．各QoSプロファイルには，前節で述べたデータプロファイルとプロトコルプロファイルが存在する．

以下に各QoSプロファイルについて説明する．

- **ユーザQoSプロファイル**
  ユーザQoSプロファイルは，ユーザの嗜好や性質を反映したQoSプロファイルである．表2.4にユーザQoSプロファイルの例を示す．

表2.4: ユーザQoSプロファイルの例

データプロファイル	英語を日本語に翻訳したい，音声で聞きたい
プロトコルプロファイル	Webクライアントで閲覧したい，メールクライアントで閲覧したい

- **アプリケーションQoSプロファイル**
  アプリケーションQoSプロファイルは，アプリケーションが利用するデータやプロトコルに関するQoSである．表2.5にアプリケーションQoSプロファイルの例を示す．

表2.5: アプリケーションQoSプロファイルの例

データプロファイル	PNG画像への対応，日本語表示の可能/不可能
プロトコルプロファイル	HTTP，POP3，SMTP，NNTP

- **デバイスQoSプロファイル**
  デバイスQoSプロファイルは，端末のハードウェアに関するQoSプロファイルである．表2.6にデバイスQoSプロファイルの例を示す．

表2.6: デバイスQoSプロファイルの例

データプロファイル	表示デバイスの大きさや色深度，音声出力の有無
プロトコルプロファイル	TCP/IP，IEEE1394，IrDA

3種類のQoSプロファイルは1つのQoSプロファイルで表現できる．例えば，ユーザQoSプロファイルが「画像データは白黒で表示したい」，アプリケーションQoSプロファイルが「GIF形式の画像データは表示可能」である場合は，画像データはGIF形式の白黒」というQoSプロファイルで表現できる．

本研究では，このように各 QoS プロファイルを 1 つの QoS プロファイルで表現することを，QoS プロファイルの合成と呼ぶ．QoS プロファイルの合成では，ユーザに近い QoS プロファイルが優先される．つまり，デバイス QoS プロファイルよりもアプリケーション QoS プロファイルが，アプリケーション QoS プロファイルよりもユーザ QoS プロファイルが優先される．

2.2.2 情報閲覧支援の種類

本研究では情報閲覧支援を，端末透過な情報閲覧支援とサービス透過な情報閲覧支援に分類した．ここでは，各分類について説明する．

端末透過な情報閲覧支援

本研究では，端末側の QoS プロファイルを考慮して情報の変換を行うことを，端末透過な情報閲覧支援と呼ぶ．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ことである．図 2.1 に端末透過な情報閲覧支援の例を示す．

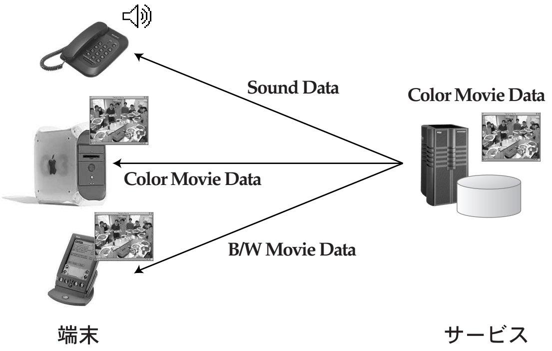


図 2.1: 端末透過

この例では，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．

サービス透過な情報閲覧支援

本研究では，サービス側の QoS プロファイルを考慮して情報の変換を行うことを，サービス透過な情報閲覧支援と呼ぶ．サービス透過とは，ユーザがサービスの特性を考慮することなく利用できることである．図 2.2 にサービス透過な情報閲覧支援の例を示す．

この例では，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．

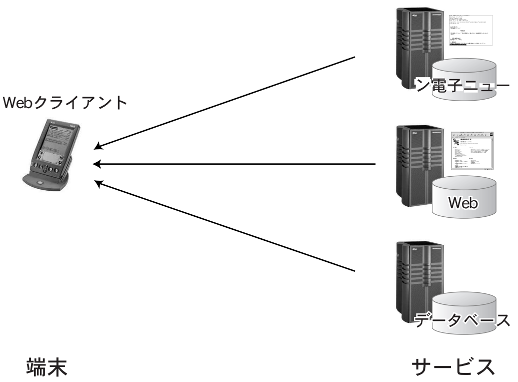


図 2.2: サービス透過

2.2.3 情報閲覧支援の必要性

ここでは，情報閲覧支援の必要性について，いくつかの例を挙げて説明する．

アクセシビリティ

アクセシビリティとは，身体障害者への支援として，ユーザの要求に適する形式で情報を提供することである．例えば，目に障害を持った人のための画像データの音声化や，耳に障害を持った人のための音声データの文字化などが挙げられる．

クライアントに適したデータの配信

クライアントアプリケーションは利用可能なデータ形式が限られているため，クライアントで対応していないデータは閲覧できないので，対応しているデータへ変換する必要がある．例えば，Web クライアントで閲覧するための TIFF 形式の画像から JPEG 形式への変換，メールクライアントのための画像からテキストへの変換などが挙げられる．

携帯小型端末への支援

携帯電話や PDA など，処理能力が低く入出力デバイスに制限がある端末で情報を閲覧するには，端末の環境に適した情報を取得するのが望ましい．例えば，表示デバイスが小さい携帯電話のための画像データの縮小や，処理能力が低い PDA のための動画から静止画への変換などが挙げられる．

非対応サービスの閲覧

通常端末には，閲覧したいサービスに対応するクライアントをインストールする必要がある．複数のサービスを閲覧する場合はそれに対応する複数のクライアントをインストールする必要がある．しかし，携帯電話などあらかじめインストールされているクライアントが変更できない端末では，閲覧できるサービスが限られてしまう．そのように，対応していないサービスをクライアントで閲覧する要求もある．例えば，携帯電話からの電子ニュースサービスの閲覧，家庭用ゲーム機での動画の受信などが挙げられる．

2.3　本章のまとめ

本論文で想定している情報環境は，コンピュータネットワークに接続された多種多様な端末とサービスで構成された環境を指す．端末の多様性は端末に適したデータを得られない，サービスの多様性はサービスを閲覧できないなどの問題を引き起こす．

本論文では，端末の性質やサービスの特性をQoSプロファイルにより表現した．また，QoSプロファイルを考慮して変換を行うことを情報閲覧支援と定義した．情報閲覧支援には，端末透過な情報閲覧支援と，サービス透過な情報閲覧支援に分けられる．サービス透過な情報閲覧支援はサービスのQoSプロファイルを考慮して情報の変換を行うことであり，端末透過な情報閲覧支援は端末のQoSプロファイルを考慮して情報の変換を行うことである．

次章では，情報閲覧支援を実現するシステムについて説明し，その問題点について述べる．

第3章　情報閲覧支援システム

本章では，情報閲覧支援システムの現状と問題点を述べる．

まず，既存の情報閲覧支援システムの分類を行い，それらの比較を行う．次に，関連研究として，プロクシモデルの情報閲覧支援システムを挙げ，説明する．また，既存の情報閲覧支援システムの問題点を挙げる．

3.1 情報閲覧支援システムの現状

情報閲覧支援システムとは，端末やサービスの QoS プロファイルに適するようにサービスで提供されている情報を変換するシステムを指す．この章では，情報閲覧支援システムの現状について述べる．

3.1.1 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，情報を変換する計算機のコンピュータ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．

3.1.2 情報閲覧支援システムの機能

情報閲覧支援システムでは，2.2.2 項で示した，端末透過な情報閲覧支援と，サービス透過な情報閲覧支援の両方を行うことが望ましい．端末透過な情報閲覧支援を行うことを端末透過性，サービス透過な情報閲覧支援を行うことをサービス透過性と呼ぶ．次にそれぞれの性質について説明する．

- サービス透過性
  サービス透過性とは，ユーザ側からさまざまなサービスが同じように利用できることを指す．サービス透過性があるシステムでは，ユーザはさまざまなサービスを一元的に閲覧できる．

- 端末透過性
  端末透過性とは，サービス側からユーザの環境が同じように見えることを指す．

端末透過性があるシステムでは，サービス提供者がユーザの環境を考慮せずに情報を提供できる．ユーザは環境に最適な情報を取得できる．

3.1.3 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もある．

サービス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モジュールを持つことになる．これは，サービスの機能肥大化を促し，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のに，サービス指向モデルではサービスの変更が，クライアント指向モデルではクライアント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．

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．

3.2 関連研究

本研究では情報閲覧支援システムにはプロクシモデルを採用する．本節では，プロクシモデルの情報閲覧支援システムの関連研究を紹介する．

3.2.1 WBI

WBI[1] は，IBM Almaden 研究所で研究された Web Intermediaries をもとに開発された Web プログラミングプロクシサーバである．

WBI ではプロクシでの処理モジュールを MEG(Monitor/Editor/Generator) と呼ぶ．MEG は，Request Editor，Generator，Document Editor，Monitor，Autonomous の 5 種類に分類される．各 MEG の入出力データは Request または Document からなる．Request はクライアントからの要求データで，Document はサーバから取得したデータである．各 MEG の説明を表 3.1 に示す．

WBI では，MEG を組み合わせることで，データ変換やキャッシュ保持，ユーザの嗜好に合わせたデータの作成などの複雑な処理が行える．

図 3.1 に WBI のシステム概念図を示す．この例では，クライアントから HTTP で要求を受けると，M(Monitor) で端末の環境情報を取得し，サーバに要求を送信する．サーバから取得したデータは，Monitor で受け取った環境情報を考慮して DE(Document Editor) で変換され，クライアントに返される．

表 3.1: WBI の MEG の分類

名前	入力	出力	説明
Request Editor	Req	Req	Request を変更する．
Generator	Req	Doc	Request に応じた Document を作成する．
Document Editor	Req/Doc	Doc	Document を変更する．
Monitor	Req/Doc	なし	Request や Document から計算を行う．
Autonomous	なし	なし	バックグラウンド処理など．

Req: Request，Doc: Document

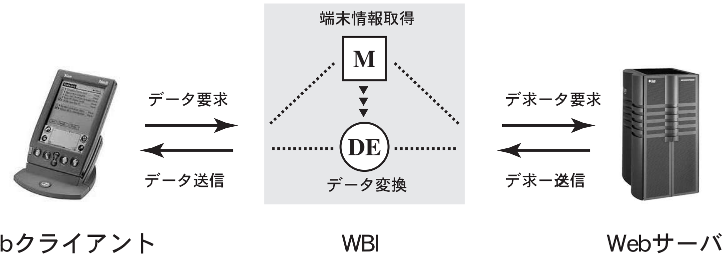


図 3.1: WBI のシステム概念図

3.2.2 KMSF-MCAP

筆者らの先行研究である Keio Media Space Family-MediaCAP(KMSF-MCAP) システムは，PDA 上での Web 閲覧を支援する目的で開発された．柔軟で自由度の高い変換処理の指定がこのシステムの特徴である．KMSF-MCAP システムの概念図を図 3.2 に示す．

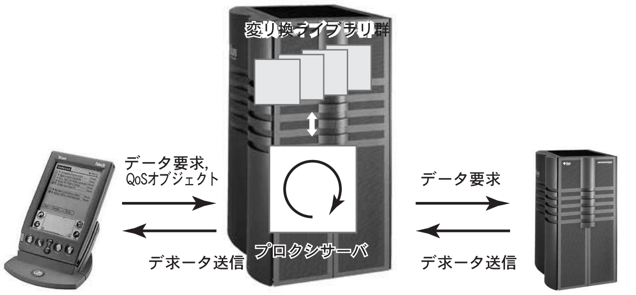


図 3.2: KMSF-MCAP のシステム概念図

このシステムは，任意の Web サーバ，3Com 社製 PalmIII 上に実装されたクライアントソフトウェア，そしてデータの中継と変換を行う MCAP サーバによって構成され

る．まずクライアントは，期待するQoSパラメータをSGML文によって記述したQoSオブジェクトを作成する．そして，MCAPサーバにQoSオブジェクトとともにURLによる情報取得要求を送信する．これを受信したMCAPサーバは，指定されたWebサーバに接続を行い，データを取得する．取得したデータにQoSオブジェクトを参照しながら加工処理を行う．この加工処理には，PDAに適する画像形式への変換，画像サイズの縮小，文字コード変換などが含まれる．これらの処理を経たデータをクライアントに提供することにより，クライアント側での処理の軽減を図る．QoSオブジェクト中には，入力データ形式と出力データ形式，加工処理に用いるパラメータなどの様々なQoSパラメータを記述できるため，ユーザの嗜好に応じた柔軟な情報閲覧が可能となる．

3.2.3 TranSend

TranSend[7]は，カリフォルニア大学バークレイ校で開発されたデータ変換Webプロクシサーバである．TranSendは，TACC[5]と呼ばれるプログラミングモデルに基づいて開発されたPythia[6]プロクシサーバをベースにしている．TACCモデルではアプリケーションは単純なAPIによって相互接続されたworkerと呼ばれるモジュールによって構成される．図3.3に，TranSendのシステム概念図を示す．

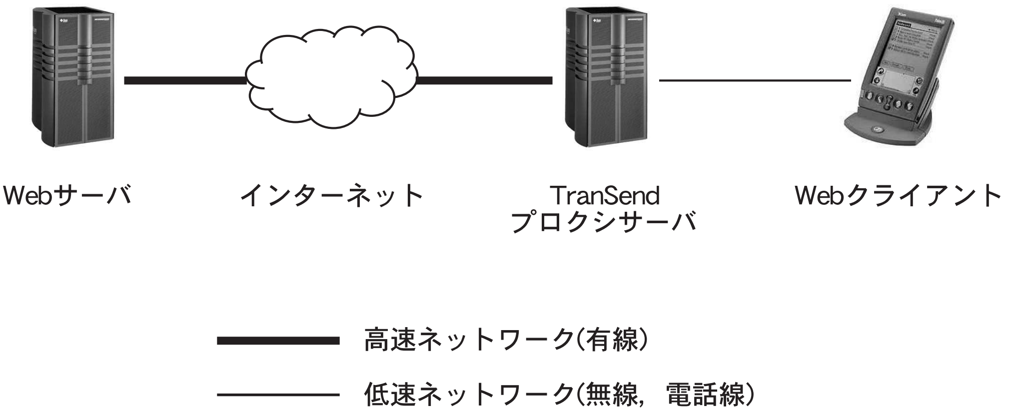


図 3.3: TranSendのシステム概念図

TranSendのworkerには，データ形式の変換を行うものや，画像の拡大縮小，解像度変換を行うものなどがある．workerの決定には，クライアントの要求するMIMEタイプによって行う．サーバから取得したデータのMIMEタイプが一致するworkerが存在する場合には，そのworkerにデータを渡し，変換を行う．一致するworkerが存在しない場合，データは変換されない．

また，TranSendではHTTPプロトコル以外にも圧縮されたHTTPプロトコルも利用できる．これにより，帯域の狭いネットワークに接続されているクライアントでWebを閲覧する場合でも，通常のHTTPプロトコルで閲覧する場合に比べて，閲覧時間が

短くなる．

3.2.4 DeleGate

DeleGate[18] は，HTTP や FTP，gopher など各種のプロトコルを中継し，ファイアウォールを越えてサーバと通信する機能を基礎として開発された．現在は様々な機能拡張がなされ，プロクシ機能に加えてキャッシュ機能，経路制御機能，セキュリティ管理機能，データに対する変換機能など様々な機能を支援している点を特徴とする．システム概念図を図 3.4 に示す

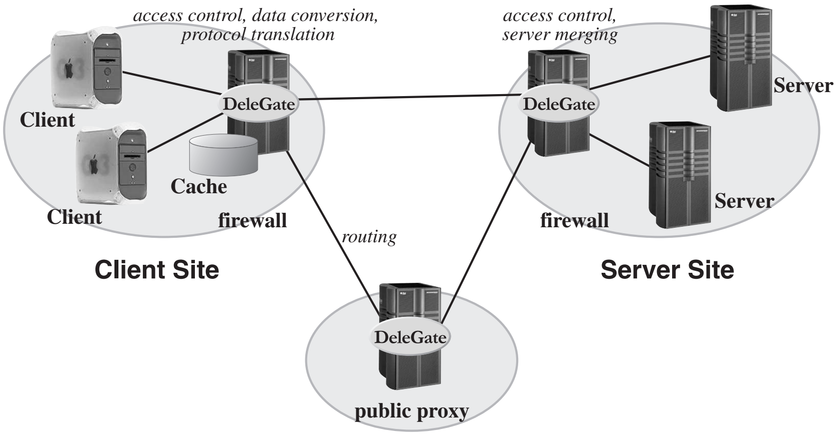


図 3.4: DeleGate のシステム概念図

DeleGate の提供するデータ変換機能は，2 種類に分類できる．一つはシステムに組み込まれたフィルタ群による変換機能である．これらのフィルタは，データに対する日本語文字コードの変換機能，文字を画像化して HTML に埋め込む機能，MIME メッセージの符号化，復号化などの機能を持つ．もう一つは，外部のフィルタプログラムを使用した変換機能である．DeleGate には CFI(Common Filter Interface) と呼ばれる外部フィルタプログラムとのインタフェイスが用意されており，これを使用すると DeleGate から任意のフィルタプログラムを起動できる．その際，DeleGate 本体，フィルタプログラム共に書き換えの必要はない．

3.3 関連研究の比較

本節では，プロクシモデルの情報閲覧支援システムで必要となる機能を分析する．以下に前節で示した関連研究の機能がどの程度実現できるかを考察し，既存研究の問題点を示す．

3.3.1 情報閲覧支援システムの付加的機能

情報閲覧支援システムが，端末透過性およびサービス透過性を提供する上で，それぞれについて実現すべき機能を次に示す．

- 動的適応性
  動的適応性とは，ユーザや端末の環境が変化したり，サービスの提供する情報が変更された場合に変換内容が変更可能なことを指す．
- 機能拡張性
  機能拡張性とは，新たな変換機能をシステムに追加できることを指す．機能拡張性があるシステムでは，新しい形式のデータや新しいサービスが登場しても，ユーザの端末に変更を加えずに閲覧できる．
- 変換連続性
  変換連続性とは，いくつかの変換機能を接続できることを指す．例えば，動画から音声への変換機能と，音声からテキストへの変換機能を連続して行うことで，動画からテキストへの変換機能を提供する．連続して変換を行うことで変換の種類が大幅に増える．

3.3.2 関連研究の比較と問題点

既存の情報閲覧支援システムを利用することにより，情報環境に適したデータ変換を実現できる．しかし，前節で述べた機能要件を全て満たすような情報閲覧支援システムは存在しない．関連研究の比較を表 3.2 に示す．

表 3.2: 関連研究の比較

サービス透過性

	WBI	KMSF-MCAP	TranSend	Delegate
動的適応性	×	×	×	×
機能拡張性	×	×	×	○
変換連続性	×	×	×	×

端末透過性

	WBI	KMSF-MCAP	TranSend	Delegate
動的適応性	△	○	○	×
機能拡張性	○	○	○	○
変換連続性	△	×	×	×

○ :実現可能，△ :一部実現可能，× :実現不可能

WBI は Web プログラミングプロクシサーバであるため，サービス透過性は考慮されていない．端末透過性については，モジュールを組み合わせるという点では，機能拡張性があるといえるが，動的適応性や変換連続性を実現するような枠組みは提供されていない．ただし WBI を拡張することで端末透過性は実現できる．

KMSF-MCAP も Web サービスに特化しているので，サービス透過性はない．しかし，ユーザや端末の環境を考慮したデータ変換を行えるので，端末透過性の機能拡張性，動的適応性は実現している．

TranSend は，KMSF-MCAP と同様にユーザや端末の環境を考慮したデータ変換を行うため，端末透過性があるといえる．また，プロトコル変換を行っているため，サービス透過性はあると言えるが，付加的機能は満たしていない．

Delegate は，プロトコル変換やデータ変換を外部フィルタを利用して変換できるので，サービス透過性，端末透過性ともに機能拡張性があるといえる．しかし，動的に環境変化を検知したり，連続して変換を行うことはできず，動的適応性，変換連続性はない．

このように，各関連研究でサービス透過性，端末透過性を完全に実現しているものはない．これらのシステムを利用して，情報閲覧支援を行う場合，実現されていない機能はプログラマが独自に実装する必要がある．したがって，端末およびサービス透過的な情報閲覧支援システムを提供する意義は大きい．

3.4　本章のまとめ

情報閲覧を支援するシステムは，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．本研究では，端末やサービスを変更することなく利用できるプロクシモデルの情報閲覧支援システムを研究対象とする．情報閲覧支援システムが提供すべき主要機能としては，端末透過性とサービス透過性がある．それぞれについて動的適応性，機能拡張性，変換連続性の機能項目を挙げた．

既存のプロクシモデルの情報閲覧支援システムとして，WBI，KMSF-MCAP，TranSend，DeleGate などがある．しかしこれらの既存システムで，端末透過性やサービス透過性を全ての項目で満たしているものはない．

第4章　情報閲覧支援システム tranServiceの設計

本章では，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．まず，本システムの設計方針，システム構成，動作手順について述べる．次にメディア変換経路の決定要因となるQoSプロファイルの設計について述べる．最後に設計方針に基づいて本システムの構成部品である，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設計を行う．

4.1 設計方針

本研究の目的は，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 システムを提供することである．システムの設計方針として，3.3.1 項で述べた情報閲覧支援システムの機能である，端末透過性とサービス透過性，動的適応性，機能拡張性，変換連続性を挙げる．また，プラットフォーム独立性も設計方針に挙げる．

- 動的適応性
  端末やユーザの要求を QoS プロファイルで表現するので，端末の環境やユーザの要求が変更された場合でも対応できる．
- 機能拡張性
  変換機能をモジュール化することにより，新たに変換機能が必要になった場合でも追加できる．
- 変換連続性
  モジュール化された変換機能は，QoS プロファイルを考慮して連続して接続できる．
- プラットフォーム独立性
  プラットフォームに依存しないモジュールは，再利用を行える．

4.2 システム構成

本説では，tranService システムの概要をハードウェア，ソフトウェア両面から述べる．

4.2.1 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 4.1 に示す．tranService システムは，端末とサービスの間にある計算機上で動作する．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を行い，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．

4.2.2 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 4.2 に各サブシステムの関係を示す．

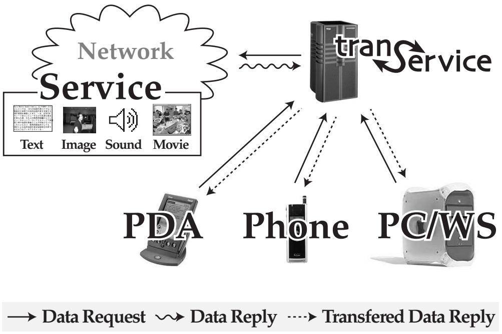


図 4.1: ハードウェア構成

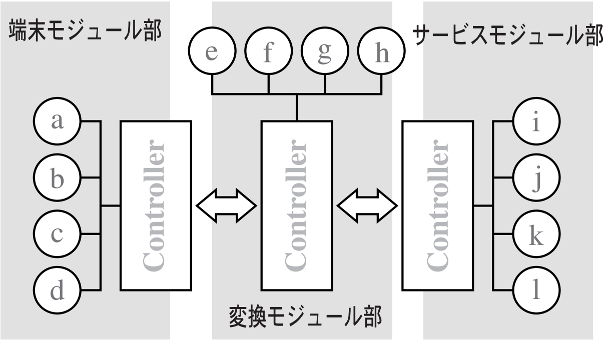


図 4.2: ソフトウェア構成

モジュール部

本システムでは，モジュール群とモジュール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
  クライアントとの通信を行うモジュール群およびモジュール群を統括するコントローラ．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
  サービスとの通信を行うモジュール群およびモジュール群を統括するコントローラ．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**
  データやプロトコルを変換するモジュール群およびモジュール群を統括するコントローラ．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ように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．

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 QoS 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．図 4.3 にデータソースの構成を示す．

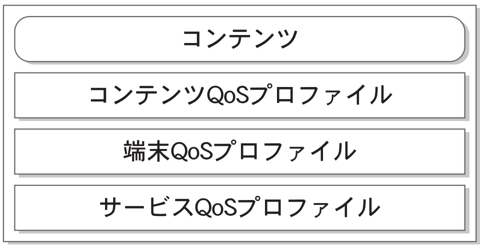


図 4.3: データソースの構成

- **コンテンツ**
  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．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作成され，変換モジュール部で変換される．

- **コンテンツ QoS プロファイル**
  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．コンテンツ QoS プロファイルも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**
  ユーザ要求や端末の環境情報．端末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 QoS 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**
  サービスからの要求やサーバの環境情報．サービス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 QoS 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

本節では，tranService システムのモジュール群の動作手順を述べる．本システムの動作手順は，クライアントからサービスへの流れと，サービスからクライアントへの流れの 2 つに分けられる．

クライアントからサービスへの流れを以下に示す．また，この手順を図 4.4 に示す．

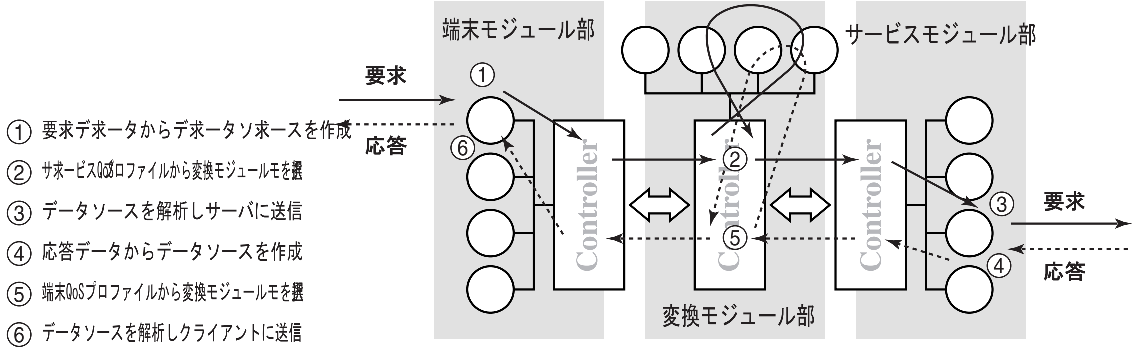


図 4.4: クライアントからサービスへの動作手順

1. クライアントが端末モジュール (a) に接続を行う．端末モジュール (a) はクライアントから要求データを受け取ると，端末 QoS プロファイルとコンテンツ QoS プロファイルを作成する．また，要求先のプロトコルに対応するサービスモジュール (k) から，サービス QoS プロファイルを取得する．これらからデータソースを作成する．
2. 端末モジュール (a) が端末モジュールコントローラに要求を渡す．
3. 端末モジュールコントローラは，変換モジュールコントローラに要求を送る．
4. 変換モジュールコントローラが適切な変換モジュール (h)(f) を選択し，データソースを順番に渡す．各変換モジュールは変換を行い，最後に変換モジュールコントローラに渡す．
5. 変換モジュールコントローラは，サービスモジュールコントローラに要求を送る．
6. サービスモジュールコントローラが適切なサービスモジュール (k) を選択して要求を送る．
7. 要求を受け取ったサービスモジュール (k) は，サービスに要求を送信する．

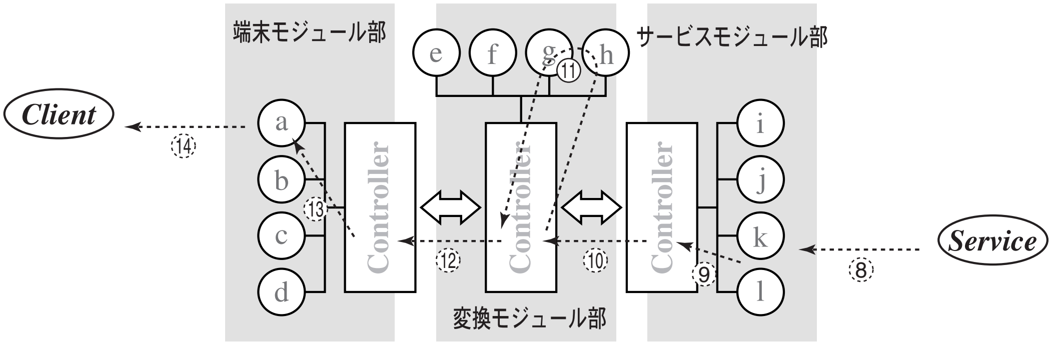


図4.5: サービスからクライアントへの動作手順

以下に，サービスからクライアントへの流れを示す．また，この手順を図4.5に示す．

8. サービスモジュール(k)が，サービスから送信された応答データを受け取る．ここで，コンテンツおよびコンテンツQoSプロファイルが作り直される．
9. サービスモジュール(k)がサービスモジュールコントローラにデータソースを渡す．
10. サービスモジュールコントローラは，変換モジュールコントローラにデータソースを送る．
11. 変換モジュールコントローラが適切な変換モジュール(g)(h)を選択し，データソースを順番に渡す．各変換モジュールは変換を行い，最後に変換モジュールコントローラに渡す．
12. 変換モジュールコントローラは，端末モジュールコントローラにデータソースを送る．
13. 端末モジュールコントローラが要求を出した端末モジュール(a)にデータソースを渡す．
14. データをソース受け取った端末モジュール(a)は，クライアントに応答データを送信する．

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．4.2.2節で述べたように，QoSプロファイルには，コンテンツQoSプロファイル，端末QoSプロファイル，サービスQoSプロファイルがある．

4.4.1 QoS プロファイルの構成

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4.6 に QoS プロファイルの例を示す．

```
QOS
├DATA
│ ├IMAGE
│ │ └TBMP
│ │   ├COLOR=1
│ │   ├DEPTH=0x02
│ │   ├WIDTH=180
│ │   └HEIGHT=180
│ └TEXT
│   └DEFAULT
│     └CHARSET=shift-jis
└PROTOCOL
  └HTTP
```

図 4.6: QoS プロファイルの例 (木構造)

データプロファイルの構成

データプロファイルは，コンテンツに関する要求や情報を格納する．データプロファイルは，コンテンツの種類 (画像や文字，音声など) によって分けられる．さらにコンテンツの種類は，コンテンツの形式 (GIF 形式や HTML 形式，AIFF 形式など) によって分けられる．コンテンツの種類および形式には，それぞれに対するユーザや端末の要求が付く．

図4.6 で示される QoS プロファイルでは，画像および文字列情報が望まれている．画像情報は TBMP 形式の白黒，色深度 2bit，最大幅 180pixel，最大高さ 180pixel が望まれ，文字列情報の形式は自由だが，文字コードは Shift-JIS エンコーディングが望まれている．

プロトコルプロファイルの構成

プロトコルプロファイルは，コンテンツを送受信する際のプロトコルに関する要求や情報を格納する．プロトコルプロファイルは，プロトコルの種類 (HTTP 形式や POP3 形式，NNTP 形式など) によって分けられる．プロトコルの種類にも，ユーザや端末の要求がつくこともある．

図4.6 で示される QoS プロファイルでは，HTTP プロトコルが望まれている．

4.4.2 QoS プロファイルの XML 表記

最初に述べたように，QoS プロファイルはコンテンツを受け取ったサービスまたは端末モジュールが作成する．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` タグの内容をみて 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したい場合は，QoS プロファイルをクライアントが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)[16] を利用している．図 4.6 で示される QoS プロファイルを XML で表記する場合は，図 4.7 に示されるようになる．

```
<?xml version="1.0"?>
 <QOS>
  <DATA>
   <IMAGE>
    <TBMP>
     <COLOR>1</COLOR>
     <DEPTH>0x2</DEPTH>
     <WIDTH>180</WIDTH>
     <HEIGHT>180</HEIGHT>
    </TBMP>
   </IMAGE>
   <TEXT>
    <DEFAULT>
     <CHARSET>shift-jis</CHARSET>
    </DEFAULT>
   </TEXT>
  </DATA>
  <PROTOCOL>
   <HTTP>
   </HTTP>
  </PROTOCOL>
 </QOS>
```

図 4.7: QoS プロファイルの例 (XML)

4.5 変換モジュール部の設計

本節では，データやプロトコルの変換を担当する変換モジュール部の設計について述べる．

4.5.1 機能

変換モジュール部は，主に以下の 2 つの機能を提供する．

- 変換モジュールの選択
  変換モジュール部は，端末モジュール部およびサービスモジュール部からデータソースを受け取ると，端末やサービスの要求に適するような変換を行うために，適切な変換モジュールを選択し，変換を行う．
- 端末モジュール部，サービスモジュール部へのデータソースの配送
  変換モジュール部がデータソースの変換を終えると，クライアントからの要求であればサービスモジュール部へ渡し，サービスからのデータソースであれば端末モジュール部に渡す．

4.5.2　変換モジュールの種類

変換モジュールには，プロトコル変換モジュールとデータ変換モジュールがある．また，データ変換モジュールは，データ種類変換モジュール，データ形式変換モジュール，データ品質変換モジュールの3つに分けられる．以下に各変換モジュールの説明を述べる．

プロトコル変換モジュール

プロトコル変換モジュールは，アプリケーションが利用するプロトコルを別のプロトコルに変換するモジュールである．例えば，Webブラウザからメールを閲覧する場合には，HTTPからPOP3へのプロトコル変換モジュールが必要になる．サービス透過性は，プロトコル変換モジュールによって実現される．

データ変換モジュール

データ変換モジュールは，コンテンツの種類または形式，性質を変換するモジュールである．以下にデータ変換モジュールの説明を行う．

- データ種類変換モジュール
  コンテンツの種類(format)の変換を行うモジュールである．例えば，文字情報から音声情報に，動画情報から静止画情報に変換を行う場合に必要となる．
- データ形式変換モジュール
  同じ種類のコンテンツのもと，コンテンツの形式(type)の変換を行うモジュールである．例えば，画像情報ではGIF形式からBMP形式に，音声情報ではWAV形式からAIFF形式に変換を行う場合に必要となる．
- データ品質変換モジュール
  同じ種類，形式のもと，コンテンツの品質(quality)の変換を行うモジュールである．例えば，画像情報の大きさや色深度，動画のフレーム数の変換を行う場合に必要となる．

4.5.3　変換モジュールの選択

ここでは，QoSプロファイルを利用した変換モジュールの選択方法について述べる．変換モジュールの選択は，変換モジュールコントローラがデータソースを受け取ったときに行う．

変換経路決定の例

例として，図4.8に示すように，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．

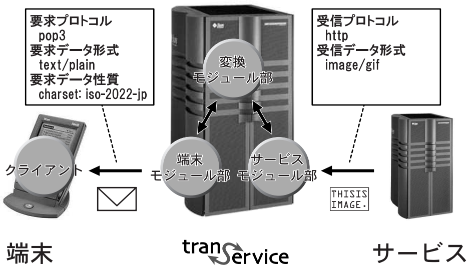


図 4.8: 変換経路決定の例

- クライアントからの要求
  クライアントが利用するプロトコルはpop3である．また，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であると仮定する．
- サービスからの要求
  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で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)であると仮定する．
- 変換モジュール
  本システムの変換モジュールコントローラには，表4.1示す4つの変換モジュールがインストールされていると仮定する．また，図4.9に，各変換モジュールの変換内容をQoSプロファイルによって表記したものを示す．

表 4.1: 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 charset:shift-jis	text/plain charset:iso-2022-jp
(d)	プロトコル変換	pop3	http

- 変換方向
  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と仮定する．

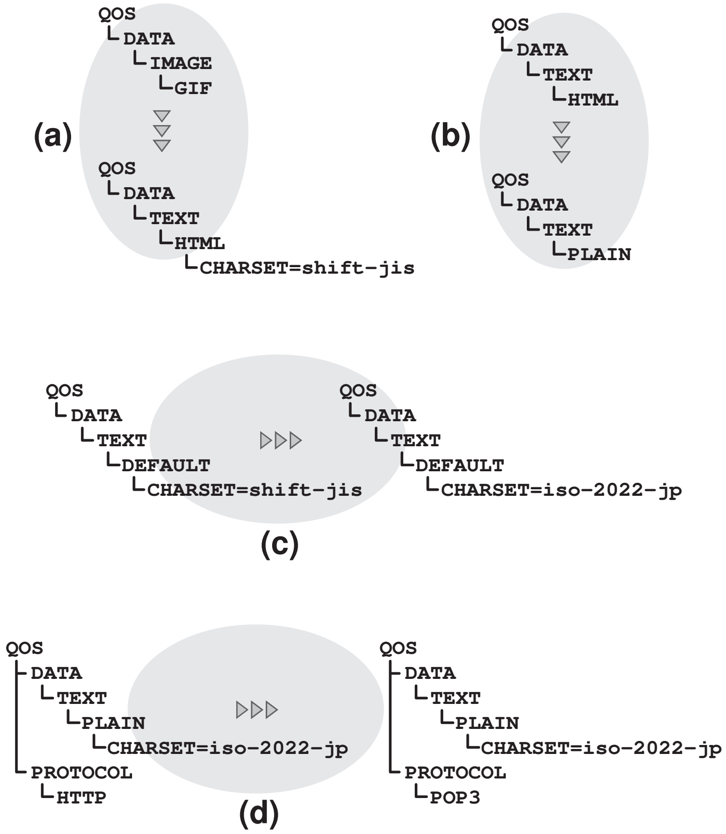


図 4.9: 変換モジュールの QoS プロファイル表記

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．

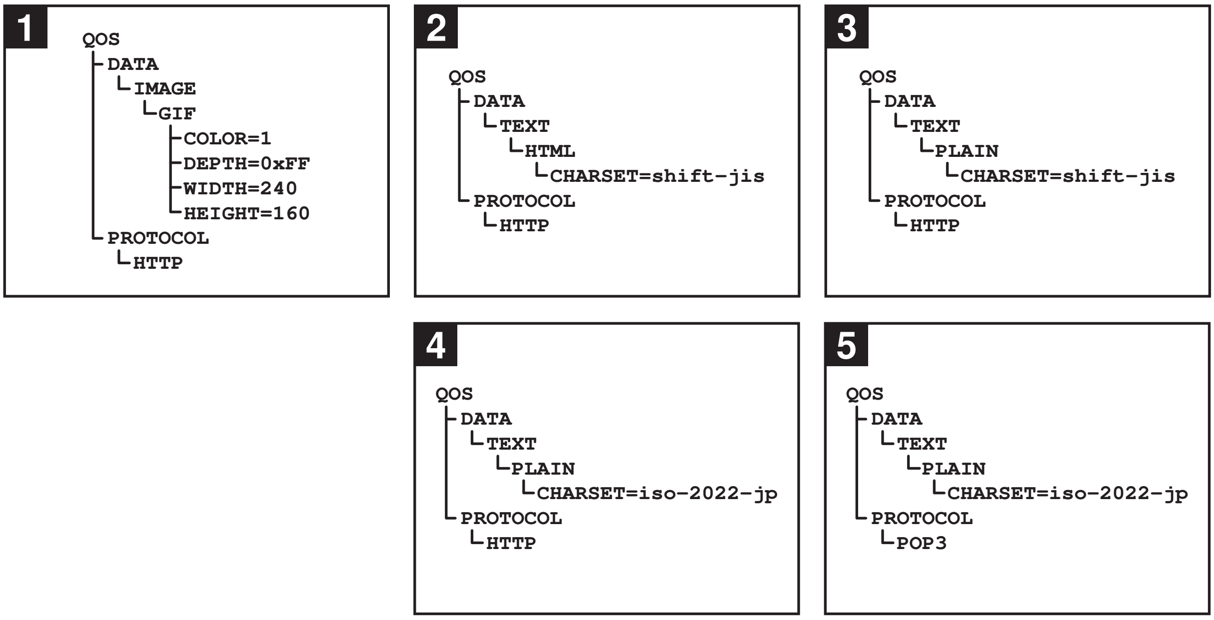
変換モジュールコントローラに渡された時のコンテンツ QoS プロファイルは，図 4.10 の 1 である．端末 QoS プロファイルは図 4.10 の 5 であるので，変換終了後にコンテンツ QoS プロファイルが端末 QoS プロファイルと同じになるように変換経路を決定する．

i. 端末 QoS プロファイル (図 4.10 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 4.10 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．



```
1
QOS
├DATA
│ └IMAGE
│   └GIF
│     ├COLOR=1
│     ├DEPTH=0xFF
│     ├WIDTH=240
│     └HEIGHT=160
└PROTOCOL
  └HTTP
```

```
2
QOS
├DATA
│ └TEXT
│   └HTML
│     └CHARSET=shift-jis
└PROTOCOL
  └HTTP
```

```
3
QOS
├DATA
│ └TEXT
│   └PLAIN
│     └CHARSET=shift-jis
└PROTOCOL
  └HTTP
```

```
4
QOS
├DATA
│ └TEXT
│   └PLAIN
│     └CHARSET=iso-2022-jp
└PROTOCOL
  └HTTP
```

```
5
QOS
├DATA
│ └TEXT
│   └PLAIN
│     └CHARSET=iso-2022-jp
└PROTOCOL
  └POP3
```

図 4.10: 変換中の QoS プロファイルの変化

4.6 端末モジュール部の設計

本節では，クライアントとの通信を担当する端末モジュール部の設計について述べる．

4.6.1 機能

端末モジュール部は，主に以下の 2 つの機能を提供する．

- クライアントとの通信
  端末モジュールは，通信プロトコルごとに用意され，クライアントとの通信を行う．クライアントからの要求データを解析し，データソースが作成される．また，変換モジュール部から受け取ったデータソースはクライアントに返される．
- 変換モジュール部との通信
  端末モジュール部で作成されたデータソースは変換モジュール部に送られる．また，変換モジュール部から受け取ったデータソースは，そのデータソースを作成した端末モジュールに送られる．

4.6.2 クライアント要求とデータソースの作成

端末モジュールがクライアントからの接続要求を受け取ると，サービスへの要求データとデータの付加情報を取得する．このとき，要求データからコンテンツとコンテン

ツ QoS プロファイルを，付加情報から端末 QoS プロファイルをそれぞれ作成する．また，クライアントが要求するサービスに対応するサービスモジュールから，サービス QoS プロファイルを取得する．その後，コンテンツおよびコンテンツ QoS プロファイル，端末 QoS プロファイル，サービス QoS プロファイルから，データソースを作成する．

4.6.3　データソース解析とクライアントへの送信

端末モジュールコントローラは，変換モジュール部からデータソースを受け取ると，要求元のクライアントと通信を行っている端末モジュールにデータソースを送る．端末モジュールは，コンテンツから応答データを作成し，クライアントに送信する．

4.7　サービスモジュール部の設計

本節では，サービスとの通信を担当するサービスモジュール部の設計について述べる．

4.7.1　機能

サービスモジュール部は，主に以下の 2 つの機能を提供する．

- サービスとの通信
  サービスモジュールは，プロトコルごとに用意され，サービスの通信を行う．変換モジュール部から受け取ったデータソースは，サービスに送られる．サービスからの応答データを解析し，データソースを作成する．
- 変換モジュール部との通信
  サービスモジュール部で作成されたデータソースは変換モジュール部に送られる．また，変換モジュール部から受け取ったデータソースからサービスへの応答データを作成する．

4.7.2　データソース解析とサービスへの送信

サービスモジュールコントローラは，変換モジュール部からデータソースを受け取ると，コンテンツ QoS プロファイルに対応するプロトコルのサービスモジュールにデータソースを渡す．サービスモジュールは，データソースを受け取ると，データソースを解析して，コンテンツから要求データを，コンテンツ QoS プロファイルから要求内容を取り出す．サービスに接続した後，情報の要求を行う．

4.7.3 サービスからの情報取得とデータソース作成

サービスモジュールがサービスから応答データを受け取ると，応答データからコンテンツおよびコンテンツ QoS プロファイルを，応答データの付加情報からサービス QoS プロファイルを作成する．その後，変換モジュール部から取得したデータソースの端末 QoS プロファイルと合わせて，データソースを作成しなおす．

4.8 本章のまとめ

本章では，tranService の設計方針として，動的適応性，機能拡張性，変換連続製，プラットフォーム独立性を実現することを述べた．次に，この設計方針に基づいて，端末モジュール部，サービスモジュール部，変換モジュール部の各サブシステムについて，提供すべき機能と実現方法について述べた．また，各サブシステム間でやりとりされる情報であるデータソースについて構成を述べ，主要構成部品の QoS プロファイルについて設計を述べた．

次章では，保証の設計を基にtranService の実装について述べる．

第5章　情報閲覧支援システム tranServiceの実装

本章では，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プロトタイプシステムの実装について詳細を述べる．

まず，本システムの実装方針について述べる．次に，各モジュール部間でやり取りされるデータソースの実装について述べる．その後で，各モジュール部の共通部の実装について述べ，変換モジュール部，サービスモジュール部，端末モジュール部の実装についても述べる．最後に，サンプルモジュールの作成方法と，実際の操作方法について述べる．

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 [9] を選択した．

Java 言語で実装したプログラムは，JVM(Java Virtual Machine) と呼ばれるインタプリタが動作する計算機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，機能拡張性があるといえる．

5.2 実装の概要

tranService の構成図を図 5.1 に示す．

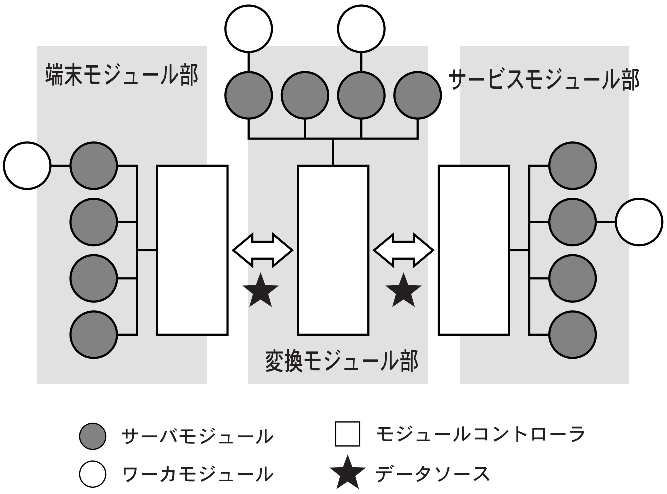


図 5.1: 実装の概要

本システムは，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により構成される．各モジュール部はモジュールコントローラとモジュール群によって構成される．モジュールコントローラ間ではデータソースの送受信が行われる．

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．

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内包している．サーバモジュールはデータを受け取ったら，ワーカモジュールを新たに作成し，データソースを渡す．

5.3　データソースの実装

本節では，各モジュール部間でやり取りされるデータソースの実装概観について述べ，次に各構成要素の実装を行う．

5.3.1　実装概観

図5.2にデータソースのクラス図を，図5.3にQoSプロファイルのクラス図を示す．

DataインタフェースはデータソースをあらわしData インタフェースを実装したものがDataImplクラスである．QosProfileインタフェースはQoSプロファイルを表し，QosProfileインタフェースを実装したものがQosProfileImplクラスである．QosProfileImplクラスは，インスタンス変数としてDataProfileクラスとProtocolProfileクラスを持っている．

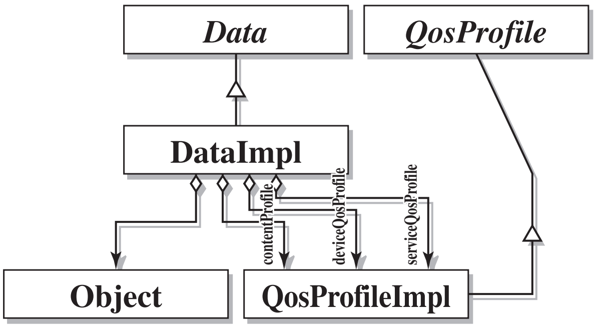


図 5.2: データソースのクラス

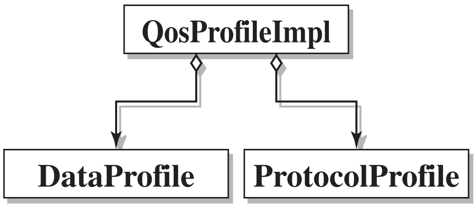


図 5.3: QoS プロファイルのクラス

5.3.2　DataImplクラス

DataImplクラスはDataインタフェースの実装クラスであり，コンテンツ，コンテンツQoS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．DataImplクラスでは，各構成要素に対する取得，設定の機能が提供されてい

る．また，作成したモジュールの識別子や，変換モジュールの順番など，データソースに対する付加情報も取得，設定できる．

5.3.3 QosProfileImpl クラス

QosProfileImpl クラスは QosProfile インタフェースの実装クラスであり，コンテンツ QoS プロファイルおよび端末 QoS プロファイル，サービス QoS プロファイルを表す．このクラスは，1 つ以上のデータプロファイルと 1 つ以上のプロトコルプロファイルから構成されており，それぞれに対する取得，設定の機能が実装されている．

5.3.4 DataProfile クラス

DataProfile クラスは，QoS プロファイルのデータに関する情報を管理するクラスである．このクラスでは，データの種類の設定および取得，付加情報の設定および取得を行える．

5.3.5 ProtocolProfile クラス

ProtocolProfile クラスは，QoS プロファイルのプロトコルに関する情報を管理するクラスである．このクラスでは，プロトコルの種類と形式の設定および取得，付加情報の設定および取得を行える．

5.4 モジュール部の実装

本節では，変換モジュール部，端末モジュール部，サービスモジュール部の各モジュール部共通の実装概観について述べ，モジュール部の構成要素の実装を行う．

5.4.1 実装概観

図 5.4 から図 5.6 にモジュールのクラス図を示す．また，図 5.7 にモジュールコントローラのクラス図を示す．

Module クラスは，全てのモジュールの基底となるクラスである．各モジュールは，Module クラスのサブクラスである，ServerModule クラスまたは WorkerModule クラスを継承して実装される．

ModuleController クラスは，全てのモジュールコントローラの基底となるクラスである．

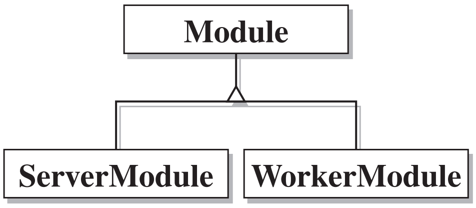


図 5.4: Module クラス

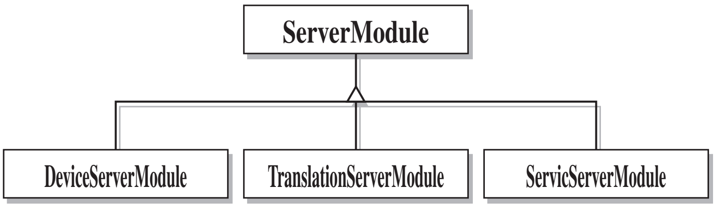


図 5.5: ServerModule クラス

5.4.2 Module クラス

Module クラスは，全てのモジュールの基底となるクラスである．次に Module クラスの機能を説明する．

- データソース転送機能
  各モジュールでは，変換や通信などの処理中にもデータソースの取得と送信が行われるように，キューを用意して非同期なデータソース取得に対応した．

  モジュールにデータソースを渡す場合は，`postData` メソッドを呼び出す．モジュールは `postData` メソッドが呼ばれると，キューにデータソースを追加する．

  キューに入っているデータソースを取得する場合は，`getNextData` メソッドを呼んで取得する．`getNextData` メソッドでデータソースを取得する際，キューが空である場合は処理がブロックされる．キューにデータソースが入るとブロックが解除され，データソースが返される．

- 付加情報の取得，設定機能

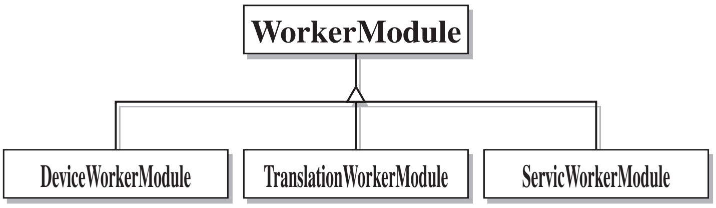


図 5.6: WorkerModule クラス

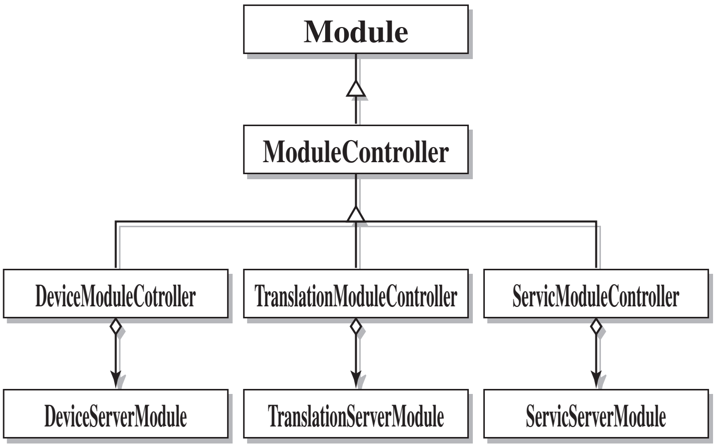


図 5.7: ModuleController クラス

Module クラスには，付加情報を取得，設定するためのメソッドが用意されている．Module クラスを継承する多くのモジュールは，この機能を利用してモジュール自体に関する情報を管理している．

Module クラスで定義されている付加情報は，クラス名，モジュール名，モジュール識別子である．

5.4.3 WorkerModule クラス

WorkerModule クラスは，クライアントやサービスとの通信，データやプロトコルの変換など，実際の処理を行うモジュールである．次に WorkerModule クラスの機能を説明する．

- データ処理機能
  ワーカモジュールで，データソースの処理を行うためには，`execute` メソッドを上書きする．`execute` メソッドは，ワーカモジュールが生成されると実行される．
- データソース取得，転送機能
  `execute` メソッドの中で `getNextData` メソッド `sendData` メソッドを呼ぶことによって，データソースの取得と転送を行う．

5.4.4 ServerModule クラス

ServerModule クラスは，1 つのワーカモジュールのクラスを保持し，そのクラスから生成したワーカモジュールのインスタンスの管理を行う．次に ServerModule クラスの機能を説明する．

- ワーカモジュールの作成機能
  ワーカモジュールの作成を要求された場合，サーバモジュールは保持しているワーカモジュールのクラスから新しいインスタンスを生成し，生成されたワーカモジュールに一意な識別子をつける．その後，生成されたワーカモジュールを，取得や削除のためにサーバモジュールのリストに加える．また，呼び出し元へ生成したワーカモジュールを返す．

- ワーカモジュールの削除機能
  ワーカモジュールの削除を要求された場合，サーバモジュールのリストに該当するワーカモジュールがある場合は，それを削除する．ワーカモジュールは，モジュール自体か識別子により指定できる．

- ワーカモジュールの取得機能
  ワーカモジュールを取得するには，モジュールの識別子を指定して取得する．この機能は，サーバモジュールでデータソースを取得した後，ワーカモジュールに渡すときに利用される．

- データソース転送機能
  モジュールコントローラから受け取ったデータソースは，識別子があれば，保持しているワーカモジュールに送る．識別子がない場合は，ワーカモジュールのクラスから新たにインスタンスを作成し，作成したワーカモジュールにデータソースを送る．

  また，ワーカモジュールから受け取ったデータソースは，対応するモジュールコントローラに送る．

5.4.5 ModuleController クラス

ModuleController クラスは，サーバモジュールの保持，管理を行う．次に ModuleController クラスの機能を説明する．

- サーバモジュールの追加機能
  指定されたサーバモジュールをモジュールコントローラに追加する．

- サーバモジュールの削除機能
  指定されたサーバモジュールをモジュールコントローラから削除する．モジュールを指定するには，モジュール名またはモジュールのクラス名を指定する．

- サーバモジュールの取得機能
  指定されたサーバモジュールを返す．モジュールを指定するには，モジュール名またはモジュールのクラス名を指定する．サーバモジュールが非活性化されている場合は，該当するモジュールがあっても何も返さない．

- サーバモジュールの活性化，非活性化機能
  指定されたサーバモジュールを活性化，または非活性化する．モジュールを指定するには，モジュール名またはモジュールのクラス名を指定する．

5.5 変換モジュール部の実装

本節では，QoS プロファイルを考慮してデータやプロトコルの変換を行う，変換モジュール部の構成要素の実装を述べる．

5.5.1 TranslationModuleController クラス

TranslationModuleController クラスは変換サーバモジュールの保持，管理を行う．

TranslationModuleController クラスがデータソースを受け取ると，コンテンツ QoS プロファイルが，目的の QoS プロファイルになるような変換モジュールの組み合わせを決定する．

変換経路の決定は，有向グラフの最短経路検索で行う．すなわち，モジュールの変換タイプ，入出力タイプをノードとし，変換モジュールはノード間の経路，コンテンツ QoS プロファイルが開始点，目的の QoS プロファイルが終了点として最短経路を探す．最短経路検索のアルゴリズムとしては，Dijkstra アルゴリズムを採用し，経路時間の短縮を行った．ただし，プロトコル変換に関しては連続変換機能を実装しなかった．

プロトコルおよびデータの変換が終了すると，端末モジュールコントローラから取得したデータソースは ServiceModuleContoroller クラスの，サービスモジュールコントローラから取得したデータソースは DeviceModuleController の `postData` メソッドを呼び，データソースを転送する．

5.5.2 TranslationServerModule クラス

TranslationServerModule クラスは変換ワーカモジュールの保持，管理を行う．

変換ワーカモジュールの管理としては，変換タイプや入出力タイプの設定，取得が挙げられる．変換タイプは，プロトコル変換とデータ変換がある．また，入出力タイプは，プロトコル変換の場合はプロトコル名，データ変換の場合は MIME タイプである．

5.5.3 TranslationWorkerModule クラス

TranslationWorkerModule クラスは実際の変換モジュールの基底となるクラスである．

また，変換の終了時には，変換後のコンテンツにあうようにコンテンツ QoS プロファイルを変更する．

5.6 端末モジュール部の実装

本節では，クライアントとの通信を担当する端末モジュール部の構成要素の実装を述べる．

5.6.1 DeviceModuleController クラス

DeviceModuleController クラスは，端末サーバモジュールの保持，管理を行う．端末モジュールからデータソースを受け取ると，TranslationModuleController クラスの `postData` メソッドを呼び，データソースを転送する．

また，変換モジュールコントローラからデータソースを受け取ると，データソースの付加情報である端末モジュール識別子から，作成元の端末モジュールにデータソースを渡す．

5.6.2 DeviceServerModule クラス

DeviceServerModule クラスは変換ワーカモジュールの保持，管理を行う．

端末サーバモジュールが，変換サーバモジュールやサービスサーバモジュールと異なる点は，クライアントからの待ち受けを行う必要があることである．そのため，端末モジュール部では，DeviceServerModule を継承したクラスを作成し，`initialize` メソッドを上書きして待ち受け処理を行う．

また，変換ワーカモジュールの管理としてプロトコルタイプの設定，取得も行う．

5.6.3 DeviceWorkerModule クラス

DeviceWorkerModule クラスは実際の端末モジュールの基底となるクラスである．

クライアントとの通信を行うモジュールは，このクラスを継承して実装する．

5.7 サービスモジュール部の実装

本節では，サービスとの通信を担当する端末モジュール部の構成要素の実装を述べる．

5.7.1 ServiceModuleController クラス

ServiceModuleController クラスは，サービスサーバモジュールの保持，管理を行う．サービスモジュールからデータソースを受け取ると，TranslationModuleController クラスの `postData` メソッドを呼び，データソースを転送する．

また，変換モジュールコントローラからデータソースを受け取ると，データソースの

付加情報であるサービスモジュール識別子から，作成元のサービスモジュールにデータソースを渡す．

5.7.2 ServiceServerModule クラス

ServiceServerModule クラスは，サービスワーカモジュールの保持，管理を行う．

また，サービスワーカモジュールの管理としてプロトコルタイプの設定，取得も行う．

5.7.3 ServiceWorkerModule クラス

ServiceWorkerModule クラスは，実際のサービスモジュールの基底となるクラスである．

サービスとの通信を行うモジュールは，このクラスを継承して実装する．

5.8 情報閲覧支援システムの利用

本節では，tranService システムを用いて情報閲覧支援プロクシサーバを構築する方法について述べる．まずはじめにモジュールの記述方法に関して，各モジュールごとに説明を行う．次に，本システムの操作方法について述べる．

5.8.1 モジュールの記述方法

ここでは，各モジュールごとに，実際の運用に必要なモジュールの記述方法について述べる．

変換モジュールの記述方法

変換モジュールは，TranslationWorkerModule を継承して実装する．

図 5.8 に `text/html` から `text/plain` に変換するモジュールの例を示す．

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の場合，変換形式は `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の場合，変換形式は `TranslationServerModule.TRANSLATION_TYPE_PROTOCOL` を指定し，入力形式および出力形式は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` メソッドが呼ばれる．データソースを受け取るには，`getNextData` メソッドを呼び，データソースを取得する．

```
import jp.ac.keio.sfc.ht.tranService.*;

public class TextHtmltoTextPlainTranslationModule
                                    extends TranslationWorkerModule {

  public TextHtmltoTextPlainTranslationModule() {
    // 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) { ... 省略 ... }
  }
}
```

図 5.8: 変換モジュールの例

データソース内のコンテンツの変換を終えたら`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．

端末モジュールの記述方法

端末モジュールは，クライアントから待ち受ける必要があるので，TranslationServerModule を継承するクラスと TranslationWorkerModule を継承するクラスを作成する．

図 5.9 にソケットを利用してクライアントを待ち受ける端末サーバモジュールの例を示す．

端末サーバモジュールが端末モジュールコントローラに追加されると，`initialize` メソッドが呼ばれる．待ち受けを行う場合は，`initialize` メソッドの中で，新たにスレッドを作成して行う．これは，データソースが非同期で渡されるため，ブロックされるのを防ぐためである．

図 5.10 に，上記のモジュールの処理部であるワーカモジュールを定義するためのインタフェースを示す．

また，図 5.11 に，HTTP プロトコルを利用するクライアントのための端末ワーカモジュールの例を示す．

端末サーバモジュールがクライアントからの接続を受け入れ，ワーカモジュールが

```
import jp.ac.keio.sfc.ht.tranService.*;
import java.net.*;

public class TcpDeviceServerModule extends DeviceServerModule {
  private ServerSocket serverSocket;

  public TcpDeviceServerModule() { }

  public void initialize() {
    int port = 8088;
    try {
      serverSocket = new ServerSocket(port);
    }
    catch(IOException ioe) { ... 省略 ... }
    // 新しいスレッドを作って，待ち受けを行う．
    new InnerThread(serverSocket).start();
  }

  private class InnerThread extends Thread {
    ServerSocket serverSocket;
    Thread thread;

    InnerThread(ServerSocket serverSocket) {
      this.serverSocket = serverSocket;
    }

    public void run() {
      while(true) {
        try {
          Socket socket = serverSocket.accept();
          // accept したら，ワーカモジュールを作成して実行する．
          TcpDeviceWorkerModule module =
            (TcpDeviceWorkerModule)makeWorkerModule();
          module.setSocket(socket);
          module.start();
        }
        catch(IOException ioe) { ... 省略 ... }
      }
    }
  }
  ... 省略 ...
}
```

図 5.9: TCP 端末サーバモジュールの例

作成されると，`execute` メソッドが呼ばれる．端末ワーカモジュールでは，クライアントから要求データを読み込み，各 QoS プロファイルを作成する．要求データと各 QoS プロファイルからデータソースを作成し，変換モジュール部に送るために，`sendData` メソッドを呼ぶ．

その後，`getNextData` メソッドでサービスから送信されるデータを待つ．データが返ってきたら，クライアントに書き込んで終了する．

サービスモジュールの記述方法

サービスモジュールは，ServiceWorkerModule クラスを継承して実装する．図 5.12 に HTTP プロトコルを利用するサービスのためのサービスワーカモジュールを示す．

```
import java.net.Socket;

public interface TcpDeviceWorkerModule {
  public void setSocket(Socket socket);
}
```

図 5.10: TCP 端末ワーカインタフェースの例

サービスサーバモジュールがデータソースを受け取り，そのデータソースに適するワーカモジュールが作成されると，`execute` メソッドが呼ばれる．サービスモジュールではデータソースを受け取った後，サービスにデータ要求を行う．

その後，サーバからデータを読み込み終えたら，データソースのコンテンツおよびサービス QoS プロファイルの変更を行う．変更を終えたら，`sendData` メソッドを呼び，サービスモジュールコントローラに渡した後，終了する．

5.8.2 プロクシサーバの操作

本システムを起動すると，変換モジュールコントローラおよび端末モジュールコントローラ，サービスモジュールコントローラが起動され，プロクシサーバを操作するためのコマンドプロンプトが表示される．システム管理者は，コマンドプロンプトに続けてコマンドを入力して，プロクシサーバの操作を行う．次に，プロクシサーバを操作するためのコマンド群を紹介する．

- **cd** コマンド
  操作対象のモジュールコントローラを指定する．引数には，指定するモジュールコントローラの名前を入力する．モジュールコントローラ名は TranslationModuleController，DeviceModuleController，ServiceModuleController またはそれらの短縮形である TMC，DMC，SMC を入力する．

  ```
  > cd [対象モジュールコントローラ]
  ```

- **add** コマンド
  操作対象のモジュールコントローラに指定されたモジュールを追加する．引数には，モジュールを示すファイルパスを入力する．また，引数に *all* を入力すると，カレントディレクトリにある全てのモジュール設定ファイルからモジュールを読み込み，追加する．

```
import jp.ac.keio.sfc.ht.tranService.*;
import java.io.*;
import java.net.*;

public class HttpDeviceWorkerModule extends DeviceWorkerModule
                                    implements TcpDeviceWorkerModule {
  public HttpDeviceWorkerModule() {
    // プロトコルは"HTTP"
    setProtocolType("http");
  }

  public void setSocket(Socket socket) {
    this.socket = socket;
  }

  public void execute() {
    try {
      // クライアントからの読み込み
      ... 省略 ...

      // 各 QoS プロファイルの作成
      Data data = new DataImpl();
      ... 省略 ...

      // サーバモジュールにデータソースを送る
      sendData(data);

      // サーバモジュールからデータソースを取得する
      data = getNextData();

      // クライアントへの書き込み
      ... 省略 ...
    }
    catch(Exception e) { ... 省略 ... }
  }
}
```

図 5.11: HTTP 端末ワーカモジュールの例

```
> add [モジュール設定ファイル]
または，
> add all
```

- remove コマンド
  操作対象のモジュールコントローラから指定されたモジュールを削除する．引数はモジュール名かモジュールのクラス名を指定する．

```
> remove [モジュール名/クラス名]
```

- list コマンド
  操作対象のモジュールコントローラに追加されているモジュールを表示する．

```
import jp.ac.keio.sfc.ht.tranService.*;
import java.io.*;
import java.net.*;

public class HttpServiceWorkerModule extends ServiceWorkerModule {
  public HttpServiceWorkerModule() {
    // プロトコルは"HTTP"
    setProtocolType("http");
  }

  public void execute() {
    try {
      // サーバモジュールからデータソースを取得する
      Data data = getNextData();

      // サーバへの書き込み
      ... 省略 ...

      // サーバからの読み込み
      ... 省略 ...

      // サーバモジュールにデータソースを送る
      sendData(data);
    }
    catch(Exception e) { ... 省略 ... }
  }
}
```

図 5.12: HTTP サービスワーカモジュールの例

```
> list
```

- enable コマンド
  指定したモジュールを活性化する．引数には活性化するモジュールのモジュール名かクラス名を指定する．

```
> enable [モジュール名/クラス名]
```

- disable コマンド
  指定したモジュールを非活性化する．引数には活性化するモジュールのモジュール名かクラス名を指定する．

```
> disable [モジュール名/クラス名]
```

- workers コマンド
  指定した MIME タイプまたはプロトコルタイプを持つサーバモジュールで動作しているワーカモジュールを表示する．

```
> workers [MIMEタイプ/プロトコルタイプ]
```

- quit コマンド
  本システムを終了する．

```
> quit
```

モジュール設定ファイル

add コマンドによりモジュールを追加する際には，モジュール設定ファイルを指定する必要がある．モジュール設定ファイルの例を図 5.13 に示す．

```
path=jp.ac.keio.sfc.ht.tranService.modules.HttpDeviceWorkerModule
server-path=jp.ac.keio.sfc.ht.tranService.modules.TcpDeviceServerModule
server-port=8088
name=Device Module for HTTP
author=Jun'ichi Yura
version=0.0.1b
enabled=true
```

図 5.13: 設定ファイルの例

`path` は，追加するワーカモジュールのクラス名を指定する．また，`server-path` には，ワーカモジュールを管理するサーバモジュールのクラス名を指定する．`server-path` が省略されると，標準のサーバモジュールが指定されたことになる．

`path` および `server-path` 以外の設定は，追加時にモジュールのプロパティとして読み込まれ，モジュールから情報を取得できる．ここでは，`server-port` でサーバモジュールの待ち受けポート番号を指定している．モジュールが追加され，サーバモジュールがソケットによる待ち受けを行う場合に，指定されたポート番号を利用できる．

5.9 本章のまとめ

本章では，情報閲覧支援システムであるtranServiceの実装について述べた．

まず，本プロトタイプシステムの実装環境を述べ，実装の概要を説明した．次に，モジュール部間でやりとりされるデータソースについて，実装概観を述べた上で構成要素の詳細を述べた．さらに，モジュール部の実装について，各モジュール部の共通の機能について述べ，各モジュール部独自の機能についても説明した．

最後に，本プロトタイプシステムのモジュール実装の方法と，操作方法について述べた．

第6章　情報閲覧支援システムtranServiceの評価

本章では，tranServiceプロトタイプシステムの評価を行う．

本章の前半では，本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う．後半では，本システムを類似システムと機能比較し，本システムが類似システムに比べて機能的に優れていることを示す．

6.1 定量的評価

本節では，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 6.1 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 6.1: 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，5.8 節で紹介した HTTP 端末モジュール，HTTP 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 6.1 の QoS プロファイルを送る．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記録するコードを埋め込み，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 6.2 に示す．また，表 6.2 に各所の説明を示す．

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．

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

図 6.1: 測定用の QoS プロファイル

表 6.2: 測定個所の説明

記号	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．

- 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，前節で説明した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 6.3 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 6.3 に示す．

(A) で処理時間が大きいのは，クライアントから送信されたデータの読み込みと，読

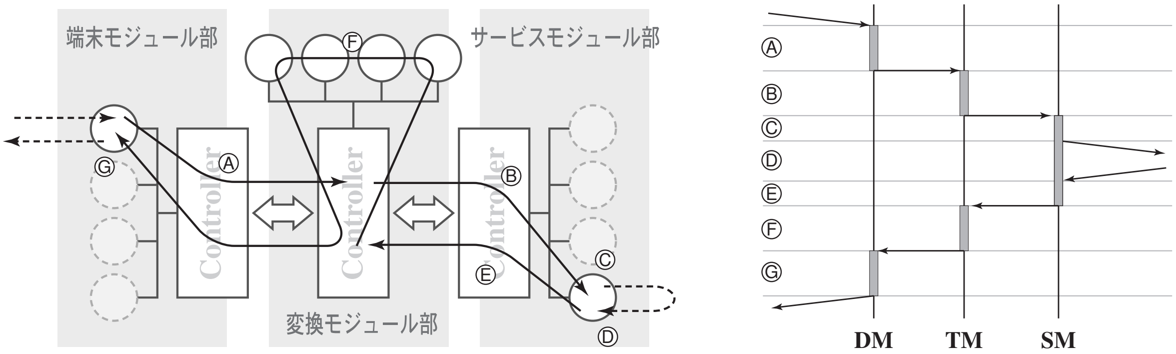


図 6.2: 測定方法

表 6.3: 測定 1 の結果

測定個所	A	B	C	D	E	F	G
測定値 (msec)	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

み込まれたデータを解析している部分に処理時間がかかるのが要因である．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がかかっている．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がかかるのが要因である．

この結果から，上の 2 点以外の処理時間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 6.4 に示す．また，測定結果をグラフ化したものを，図 6.4 に示す．

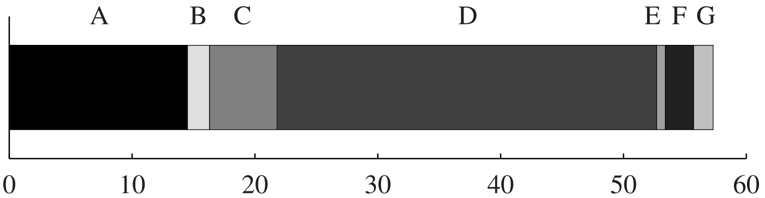


図 6.3: 測定 1 のグラフ

表 6.4: 測定 2 の結果

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

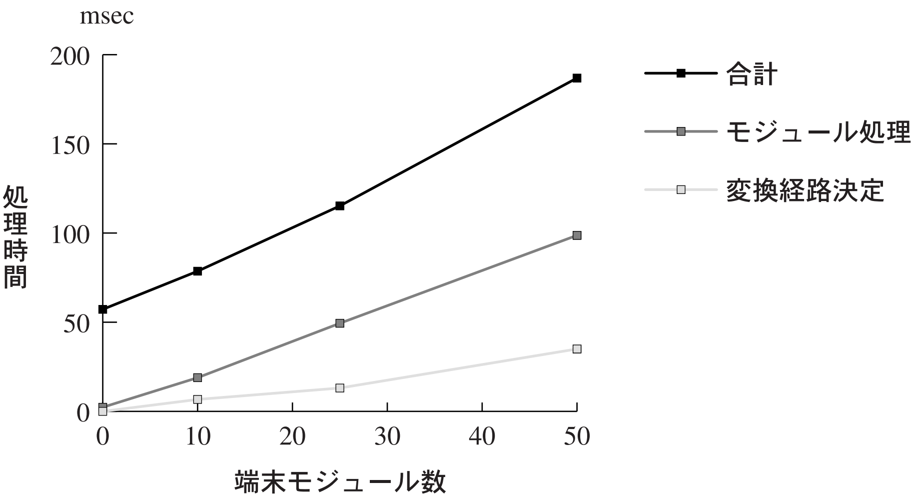


図 6.4: 測定 2 のグラフ

この結果を線形近似によって数式化したものを式 6.1 から式 6.3 に示す．

$$T_{sum} = 2.6Num + 54 \tag{6.1}$$

$$T_{exec} = 1.9Num + 1 \tag{6.2}$$

$$T_{calc} = 0.7Num - 1 \tag{6.3}$$

式 6.1 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 6.2 はモジュール処理時間 (T_{exec}) を，式 6.3 は変換経路決定処理時間 (T_{calc}) を表したものである．

式 6.3 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 6.2 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，3.2 節で挙げた類似システムと，tranService との機能評価を行う．評価項目および評価結果を表 6.5 に示す．

表 6.5: 定性的評価

サービス透過性

	WBI	KMSF-MCAP	TranSend	Delegate	tranService
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

	WBI	KMSF-MCAP	TranSend	Delegate	tranService
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	○	○	×	×	○

○ :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，将来的には実装する予定である．

- 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，サービス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わなく

てはいけない．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．

6.3 本章のまとめ

本章では，tranService プロトタイプの測定および評価を行った．

本章の前半では，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の基本的性能特性を把握するために定量的評価を行った．システム内部のオーバヘッドとしては，ネットワークからの読み書きや XML の変換部分で多少の処理コストがあったが，システム全体から見ると支障をきたすものではなかった．また，モジュール数の増加に伴い，変換経路決定の処理時間も比例して増加したが，改善できることも示した．

本章の後半では，類似システムとの機能を比較し，本システムが類似システムに比べて優れていることを示した．

第7章　結論

7.1　まとめ

本論文では，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，簡単に本システムを利用できる．

情報閲覧を支援するシステムは，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．本研究では，端末やサービスを変更することなく利用できるプロクシモデルを採用した．情報閲覧支援システムでは，端末透過性とサービス透過性を実現する必要がある．端末透過性はサービス側からユーザの環境が同じように見えることであり，サービス透過性はユーザ側からさまざまなサービスが同じように利用できることである．情報閲覧支援システムの提供すべき機能として，端末透過性とサービス透過性のそれぞれに動的適応性，機能拡張性，変換連続性がある．既存のプロクシモデルの情報閲覧支援システムでは，それらの機能が実現されていないか，実現されていても一部しか満たしていなかった．そこで本システムでは，以上の機能の実現を目標とした．

本研究では，端末透過性とサービス透過性を実現するtranServiceシステムを開発した．本システムは，データやプロトコルの変換を行う変換モジュール部，クライアントとの通信を行う端末モジュール部，サーバとの通信を行うサービスモジュール部から構成される．各モジュール部間では，データソースと呼ばれるコンテンツやQoSプロファイルからなる情報がやり取りされる．QoSプロファイルはユーザが要求する情報に対する制約や，サービスの特性によって生じる制約に関する情報である．本システムでは，サービスへの要求の場合は，クライアントからのデータをサービスのQoSプロファイルに適するように変換する．また，クライアントへのデータ送信の場合は，端末のQoSプロファイルに適するように変換する．このようにすることで，端末透過性およびサービス透過性を実現している．

また，本システムでは動的適応機能や機能拡張機能，連続変換機能を提供している．クライアントやサーバからデータを取得する際，データの付加情報などからQoSプロファイルを生成することで動的適応性を実現する．また，各モジュールは，本システム起動後も，適宜追加や削除が行えるという点で機能拡張性がある．また，QoSプロファイルをもとに変換を行う際，複数のモジュールを組み合わせることによって連続変換性も実現した．

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，モジュールのネットワーク透過機能を実現することにより，負荷分散や耐故障性も向上できると思われる．

7.2 今後の課題

tranServiceシステムの今後の課題としては，以下の4点が挙げられる．

連続メディアデータへの対応

現在，データソースのコンテンツはサービスから全て読み込み，静的なデータとしてメモリ内に作成する．これは，モジュールで変換する場合，コンテンツの大きさや種類を特定できる静的なデータの方が扱いやすいからである．しかし，動画のストリーミング再生などを行う場合には静的なデータでは対応できないため，コンテンツをストリームとして持つなどの実装を行う必要がある．

モジュールのネットワーク透過機能

現在の実装では，各モジュールおよびモジュールコントローラは同一のJVM上で動作する必要がある．しかし，変換処理の効率やモジュールの共有の面からすると，ネットワークを介した複数の計算機のJVM上で動作するモジュールが連携できることが望ましい．そこで，モジュールをネットワークを介して利用できるように，モジュールコントローラの機能を拡張することで，ネットワーク透過なモジュールの利用を提供する．

QoSプロファイルの互換性の向上

本システムは，端末やサービスの要求をQoSプロファイルを用いて表すことで，変換経路決定機能を実現している．端末やサービスの要求を表すには，MPEG7[11]，端末通知プロトコル[10]，CC/PP(Composite Capability/Preference Profiles)[14]など，既存の仕様が存在している．既存仕様とQoSプロファイルを相互に変換することで，その仕様を利用するクライアントやサーバに容易に対応できる．

システム全体のパフォーマンス

6.1.3節で述べたように，システム全体のパフォーマンスを向上させるには，変換経路決定の処理時間を短縮する必要がある．そのためには，変換経路決定アルゴリズム

の改善や，キャッシュ機構の追加などが有効である．

また，サービスから取得したコンテンツや，変換後のデータなどもキャッシュすることによって，システム全体のパフォーマンスを向上できると予測される．

謝辞

本研究を進めるにあたり，絶えず懇切丁寧な御指導を頂きました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝いたします．

慶應義塾大学徳田・村井・楠本・中村研究会の諸先輩方には，折りにふれ貴重な示唆や御助言、御指導を頂きました．特に慶應義塾大学政策・メディア研究科修士2年の大越匡氏，中澤仁氏には，本論文執筆にあたって絶えざる励ましと御指導を頂きました．慶應義塾大学政策・メディア研究科修士2年の坪井孝弘氏，修士1年の牧野聡氏，藤村浩士氏には，多忙の中熱心な指導とさまざまな助言を頂きました．ここに深い感謝の念を表します．

また，Keio Media Space Family-Architecture(KMSF) 研究グループのメンバーには，本研究に関する様々な議論をして頂きました．西尾信彦氏，戸辺義人氏をはじめとする慶應義塾大学 MKG プロジェクトの皆様には，本研究を進めていく上でのさまざまな助言を頂きました．ここに深い感謝の念を表します．

最後に，研究生活を日夜を問わず共に過ごした新井仁氏，楠本晶彦氏，中村勇樹氏，西野秀明氏，原嶋章介氏，その他多くの友人たちに深く感謝し，謝辞と致します．

平成12年1月31日
慶應義塾大学 環境情報学部4年
由良 淳一

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] F501i. http://www.fujitsu.co.jp/hypertext/Products/telcom/mobileworld/product/.

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

[5] Armando Fox. The Case For TACC: Scalable Servers for Transformation, Aggregation, Caching, and Customization. Qualifying Exam Proposal.

[6] Armando Fox and Eric A. Brewer. Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation. In *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 1996.

[7] 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.

[8] IntelliStation. http://www.ibm.co.jp/pc/intellistation/product.html.

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

[10] Tomihisa Kamada and Tomohiko Miyazaki. *Client-Specific Web Services by Using User Agent Attributes*, 1997. http://www.w3.org/TR/NOTE-agent-attributes.

[11] Jose M. Martinez. *Overview of the MPEG-7 Standards*, 1999. ISO/IEC JTC1/SC29/WG11 N3158.

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

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

[14] Franklin Reynolds, Johan Hjelm, Spencer Dawkins, and Sandeep Singhal. *Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation*, 1999. http://www.w3.org/TR/NOTE-CCPP/.

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

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

[17] WorkPad. http://www.ibm.co.jp/pc/workpad/.

[18] 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.