モーバイルコンピューティング 7－1
（1998. 12. 4）

LDAP を用いたアプリケーションプログラム自動設定ツールの設計と実装

後藤 真孝 † 尾崎 哲 †

ネットワークにまつわるアプリケーションプログラムは，ネットワーク環境の状態の変化に伴い，設定の変更を余儀なくされる場合が多い．そこで，我々は，移動計算機が LDAP を用いてディレクトリサーバから自機のアプリケーションプログラムの設定情報を取得し，移動のたびに必要な設定変更を自動化するツールの設計と実装を行なっている．

本稿では，LDAP を用いた自動設定の実現のための基本的な課題を検討し，(1) 設定情報のディレクトリへの保存方法，(2) アプリケーションプログラムの設定変更方式という 2 つの課題の解決方法を示す．また，試作した自動設定ツールについて述べる．本ツールにより，移動時の設定変更に関してユーザの手を煩わせることなく自動化することが可能となる．

Design and Implementation of LDAP based Auto-configuration Tools

MASATAKA GOTO † and SATOSHI OZAKI †

When we use network based application programs, we often need to perform re-configuration of various parameters due to changing the network environment. That is the motivation of our design and prototyping of auto-configuration tools, which collect the necessary configuration information for those application programs from a directory server using LDAP convention, then setup it on the application programs suitably.

In this paper, we examine some primitive issues on implementing the LDAP based auto-configuration tools, and show solutions of the following issues: (1) how to maintain the various configurations in a directory, (2) how to change the configuration of individual application program. And we also presents how we designed and prototyped our tools. These tools make it possible for some popular network application programs to be available with easy setup operations.

1. はじめに

低価格化による普及により，パーソナルコンピュータ (PC) は様々な業種のオフィスで使用されるようになった．その結果，現在の企業の日常業務にとって PC は不可欠なものとなっている．また，それぞれの企業でイントラネットと呼ばれる計算機ネットワークを構築していることに象徴されるように，通信網の整備と普及が進みつつある現在では，計算機ネットワークも日常的で不可欠なものとなりつつある．

一方，PC の普及の結果，PC に精通していないユーザの増加が進んでいる．日常業務で必要となる既存のアプリケーションプログラムにおいてさえ，設定にはある水準の知識や経験が必要であり，設定作業は，PC に精通していないユーザがアプリケーションプログラムを正しく動作させることができない 1 つの要因となり得る．

また，PC の小型化と軽量化が進み，自分用の PC を持ち歩き，移動先でその PC を使用する形態も普及してきた．しかし，移動先ではネットワーク環境を利用する際，移動に伴う設定変更作業が必要な場合があり，これは，PC に精通しているユーザにとっても面倒な作業となり得る．また，使っているネットワーク環境側の変更に伴う設定作業が必要な場合もある (例えば，プリンタの置き換えといったもの)．いずれの場合においても，設定工数は無視できるものではない．

ところで，情報の配送と管理の技術にディレクトリと呼ばれるものがあり，ITU-T ☆ で標準化されている．ディレクトリでは，データベースから統一手順で情報取得ができるようになっており，多種多様な情報の管理と配送に応用できる枠組になっている．また，イン

† 株式会社 東芝 研究開発センター 情報・通信システム研究所
Communication and Information Systems Research Labs., Research and Development Center, Toshiba Corporation

☆ The Telecommunication standardization sector of International Telecommunications Union

ターネットプロトコルでの使用を前提としたディレクトリをアクセスするためのプロトコルである LDAP (Lightweight Directory Access Protocol) が IETF ☆ で標準化されつつある.

そこで我々は, ディレクトリによる情報配送サービスの応用の 1 つとして, アプリケーションプログラムの設定情報の配送にディレクトリを用い, ユーザの設定作業を自動化する方式の設計と試作を行なっている. 本稿では, LDAP を利用したアプリケーションプログラムの自動設定方式について説明する. そして, 試作した LDAP API とプロトタイプ LDAP サーバ, LDAP 自動設定クライアントについて述べ, 最後に残された課題を挙げる.

2. アプリケーションプログラムの自動設定

本章では, 我々の提案するアプリケーションプログラムの自動設定について述べる. 始めに, 現状の PC の利用形態と LDAP によるアプリケーションプログラムの自動設定の必要性について述べる. 次に, 必要となる機能を挙げ, 実現方針を述べる.

2.1 PC の利用形態と自動設定の必要性

ネットワークサービスを活用したアプリケーションは, 各々のネットワークで固有の環境情報を基にした設定を必要とするのが一般的であるため, 自分の PC を移動先で利用する場合には, 再設定作業を行なわなければならない. また, ネットワーク対応の設定情報は, 当該ネットワークの管理者に問い合わせなければ取得できない場合も多く, 利用者の設定のたびに管理者へ負担がかかる場合もある. 以上のように, (主に, ネットワークサービスを利用する) アプリケーションの設定には, 利用者と管理者の工数が必要であるため, 設定の自動化の需要が潜在していると言える.

アプリケーションプログラムの自動設定に応用できる技術として, 次のものの提案や標準化がなされてきた.

(1) Application Configuration Access Protocol (ACAP)[3]

(2) Service Location Protocol (SLP)[5]

(3) Dynamic Host Configuration Protocol (DHCP)[7]

表 1 に, それぞれの技術について, 特徴と自動設定における長所と短所を述べる.

ACAP は, IETF で標準化が行なわれているプロトコルで, 名前が示す通り, アプリケーションの設定を行なうためのプロトコルである. 階層化された data class と呼ばれるエントリが階層化された属性を持ち, それぞれの属性の値として設定情報を保持する. それらのエントリを持つサーバを用意し, ユーザの利用するアプリケーションがサーバから設定情報を取得する手順を定めたプロトコルである. 設定情報の取得のための認証やアクセスコントロールリストに関する枠組も用意されている. 一方, 実装環境が充分に公開されているとはいえず, また, 充分な利用実績があるとはいえない. ACAP 専用のサーバが必要であることも, 普及の妨げの 1 つの要因といえる.

SLP は, IETF で標準化が議論されているプロトコルで, ネットワークに接続された機器が, 自分の持っている能力や提供できるサービスを周りに伝え, 周りからアクセスされるようにすることができるプロトコルである. また, ディレクトリエージェントと呼ばれる SLP のサーバが, ネットワーク上の機器の能力やサービスを記憶し, 記憶した情報をネットワーク上の機器に提供する枠組もある. 一方, SLP は企業内で様々な情報の一元管理の一環として, 設定情報を管理したい場合には充分に応えられるとはいえない. また, ACAP と同様に, 実装環境が充分に公開されているとはいえない.

DHCP は, TCP/IP 上のホストにまつわる設定を自動化する技術である. これまでにも, DHCP のメッセージを拡張して, デフォルトルート情報やネームサーバ情報を配送してきた. しかし, DHCP の配送する情報は, ネットワークのセグメントで共通の情報, またはネットワークインタフェース対応の情報しか配送できない. つまり, 個々のユーザ対応の設定を配送できないため, アプリケーションの設定に用いるには機能的に制限が大きい.

そこで我々は, 設定情報だけに特殊化を必要とせず, 様々な情報管理の一環として設定情報を扱い, 管理の分散といった拡張性にも応えることのできる枠組であるディレクトリ, および LDAP に着目し, LDAP によって設定情報を配送するアプリケーションプログラムの自動設定方式 (以降, LDAP 自動設定と呼ぶ) の設計と試作を行っている. 具体的には, 会社内やネットワーク内での共通設定情報や推奨設定情報, ユーザ固有の設定情報をディレクトリサーバに保存しておき, ユーザの PC が LDAP により自律的に設定情報を取得して設定を行なうことを目指した方式である.

2.2 LDAP 自動設定の特徴と機能

設定情報の配送に LDAP を使用していることによる, 特徴は次の通りである.

(特徴 1) オープンなプロトコルである LDAP によ

☆ The Internet Engineering Task Force

表 1　自動設定に応用できる技術の特徴と長所と短所
Table 1　Advantages and disadvantages of each technology for auto-configuration.

プロトコル	特徴	長所	短所
ACAP	サーバ/クライアント型で，設定情報を保持するサーバから設定情報を取得する．アプリケーションプログラムの設定の自動化を目的として策定されたプロトコルである．	認証とアクセスコントロールリストの枠組がある．	充分な実運用がなされているとはいえない．ネットワーク上に専用サーバが必要である．実装環境の公開が不十分である．
SLP	ネットワーク上の各々の機器が，自機のサービス提供内容を周辺に知らせる．または，各々のサービス内容をディレクトリエージェントに保持しておき，ディレクトリエージェントが周辺に知らせる．	機器が自律的に周辺にサービス内容 (設定情報) を配送する．	多数のネットワーク環境の設定情報を一元管理するには向かない．ユーザ対応の設定情報の管理には向かない．実装環境の公開が不十分である．
DHCP	リンクレベルでの通信段階から設定情報やブートプログラムの配送ができる．IP アドレスの自動割り当てに広く使われている．	既に多くのネットワークで実運用されている．	無数にあるサービスやアプリケーションの設定情報への対応には向かない．MAC アドレス毎の管理が前提であり，ユーザ単位での設定情報の配送は難しい．

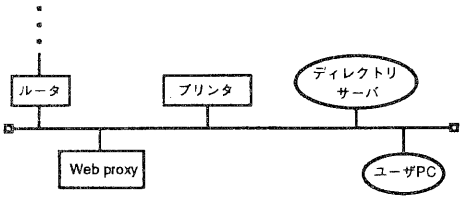

図 1　LDAP 自動設定環境の簡単な例:ネットワーク上の資源である Web proxy を使いたい場合，ディレクトリに置かれているアドレスやポート番号，プロトコル種別を含む設定情報を LDAP で取得し，ユーザ PC に設定する自動設定クライアントを実行することで，設定作業を完了する．

Fig. 1　Simple example of LDAP based auto-configuration system.

る，様々なプラットフォームへの対応

(特徴 2)　ディレクトリとデータベースの親和性

(特徴 3)　標準インタフェース仕様の存在と，LDAP ライブラリの存在

LDAP は IETF で標準化が行なわれている技術なので，プラットフォームによらない同一サービスの提供が可能である．更に，設定情報を記憶しておくディレクトリは，既存のデータベースで LDAP をサポートしたものを利用することで，既に構築されたデータベースと統一的で簡便な管理を実現することもできる．また，実装の際の課題となる API (Application Programming Interface) は C 言語[2] と Java 言語に関してほぼ仕様が決定され，C 言語と Java 言語，perl 言語の SDK (Software Developers Kit) ☆ も公開されており，実装のための現実的な環境がおおむね整ってきた．

次に，LDAP 自動設定の簡単な例を図 2.2 に示し，LDAP 自動設定が効果的に機能する具体的な状況の例を挙げる．

(状況 1)　初めてユーザ PC (クライアント) をネットワークに接続した場合の初期設定の自動化

(状況 2)　ノウハウが必要なアプリケーションの推奨設定の周知・徹底

(状況 3)　新たな機器やサーバソフトウェアが導入された時のユーザ PC (クライアント) の設定更新の自動化

(状況 4)　一時的な利用者である，来賓や来客が PC を持参し，そのネットワーク資源を利用する際の設定の自動化

上記のように，上記のように，ネットワークにまつわる設定情報だけでなく，ネットワークとは無縁のアプリケーションに関しても，管理者による推奨設定といった情報を提供し，自動的に反映することが可能である．また，一般的な計算機利用においてしばしば必要となる作業を自動化することに意義があると，我々は考えている．また，PC のユーザだけでなく管理者への負担を削減する点も重要である．更に，認証機能によりユーザを識別できるので，ユーザ個別の対応が可能である点が LDAP を使用している特徴の一つと捉えている．

上記のように，ネットワークにまつわる設定情報だけでなく，ネットワークとは無縁のアプリケーションに関しても，管理者による推奨設定といった情報を提供し，自動的に反映することが可能である．

以上から，具体的な LDAP 自動設定で必要となる機能には，次のものがある．

(機能 1)　アプリケーションプログラムの設定に必要な情報の配送と取得

(機能 2)　ユーザ PC のアプリケーション設定への自

☆ http://developer.netscape.com/tech/directory/index.html

動反映

(機能 3) ディレクトリ機能としての認証やアクセス制御による，共通設定だけでない (例えば，個々のユーザ対応や現在位置に応じた) 設定情報の提供

(機能 4) 設定情報の一元管理と分散管理の実現

(機能 5) ユーザ PC やユーザの認証，通信内容の暗号化によるセキュリティの確保

自動設定に必要な最低限の機能は，(機能 1) と (機能 2) である．その他の機能は，利用形態に併せた高機能化の一環と捉えることができる．そこで今回は，必要最低限である (機能 1) と (機能 2) を実現することとした．

2.3 LDAP 自動設定の設計

今回の LDAP 自動設定の実現課題を，次に示す．

(課題 1) 設定情報のディレクトリへの保存方法

多様なアプリケーションプログラムがあり，それぞれのアプリケーションプログラム毎に必要な多様な設定情報を，ディレクトリエントリとして保存しておく方法．

(課題 2) ディレクトリからの取得方法とアプリケーションの設定変更

ディレクトリサーバの探索，ディレクトリから設定情報を取得する手順，取得した設定情報の反映の方法．

今回は，次の方針に従い上記の課題を解決し，試作を実装した．

(方針 1) ネットワークでのサービス (プロトコル) に着目する

アプリケーション対応に設定情報を扱うのではなく，同種のネットワークサービスを扱うアプリケーションに共通の設定を自動化した．現在，様々なベンダーから製品化されたアプリケーションは無数にあり，今後も，膨大な数のアプリケーションが製品化されるであろう．この場合，各製品対応に設定情報を提供するのは効率的ではない．また，頻繁な設定変更を余儀なくされるのはネットワーク間を移動する移動端末であり，その場合の変更内容はネットワークにまつわる設定である．従って，この前提は LDAP 自動設定の最も効果的な使用用途の 1 つを扱っているといえる．

(方針 2) ユーザ PC 側の自動設定を行なうプログラムは，サービス (プロトコル) 毎の設定情報の検索の方法を知っている

ユーザ PC がディレクトリに対してどのような問い合わせをすべきかを事前に知っている，つまり，欲しい設定情報の取得方法を知っていることとした．これは，自動設定のモデルを不用意に複雑化させないためである．ネットワークサービスの設定情報の取得へ動的に対応する機能は，残された課題の 1 つとする．

上記の方針の下，(課題 1) と (課題 2) を次のように解決した．

2.3.1 (課題 1) 設定情報のディレクトリへの保存方法への対処

(方針 1) により，設定情報はサービス (プロトコル) 単位で分類できる．そこで，プロトコル単位で設定情報をディレクトリエントリとする．各々のサービス (プロトコル) での設定内容は，そのディレクトリエントリの属性として保持する．

例えば，WWW (HTTP) の設定情報は CN=HTTPproxy というディレクトリエントリを作成し，その属性として host=proxy.toshiba.co.jp と port=8080, noProxy=*.toshiba.co.jp を保持する．

2.3.2 (課題 2) ディレクトリからの取得方法とアプリケーションの設定変更への対処

(方針 2) より，ユーザ PC は欲しい設定情報の取得方法は事前に知っている．それぞれのアプリケーションが，各自で自律的に設定情報を取得する方法も考えられるが，既存のソフトには適用できない上に，これから作成されるソフトが各々で LDAP を使用して取得するのは生産的とはいえない．そこで，LDAP で設定情報を取得し，様々なアプリケーションプログラムに反映させるプログラム (LDAP 自動設定クライアントと呼ぶ) により自動設定を実現する．ここで，アプリケーションの設定情報は，ディスク上のファイルやレジストリといった形態で保存されていることを仮定している．この LDAP 自動設定クライアントを用いた形式における，設定情報の保存形態や書式に関しての動的な対応についての検討は，残された課題の 1 つとする．また，LDAP サーバの発見には，文献 4) で提案されている方法の 1 つである，ldap という名前のホストをドメインを遡りながら探す方法を採用する．

3. LDAP API の実装

LDAP の実装において，サーバ/クライアントで用いるライブラリ関数群を，FreeBSD 2.2.6 上と Windows95/98/NT4 上に実装した☆．サーバおよびクライアントは，それぞれの LDAP API を介してライブラリ関数を用いることで，LDAP の実装を容易に行

☆ 本試作の実装開始の時点では，LDAP SDK の公開は行なわれていなかった．

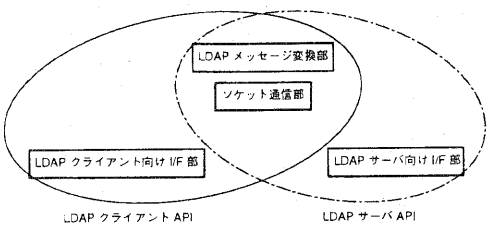

図 2 LDAP API の構成：LDAP メッセージ変換部は，BER (Basic Encoding Rules) でエンコードされた LDAP メッセージとプログラム中で扱う構造体の相互変換を行なう．ソケット通信部は，TCP/IP のインタフェースである socket を使用し通信する関数群である．LDAP クライアント向け I/F 部は，クライアント API として提供するインタフェース関数である．サーバも同様である．

Fig. 2 Structure of LDAP API.

なうことができるようになる．

LDAP API の構成を図 2 に示す．日本語サポートの補足的な機能として，LDAP メッセージ変換部では JIS コードと UTF8☆ の変換を行なっている．これにより，LDAP メッセージでは UTF8 を使用することになっているが，この API を用いた開発環境では，より扱い易い JIS コードを用いることができるようになる．

3.1 クライアント API

実装したクライアント API の仕様は文献 2) に従っており，必要最低限の機能を実装した．

クライアント API は，典型的には，次の 4 つのステップで使用する．

(1) LDAP セッションを初期化する．ldap_init() を呼び，セッションのハンドルを戻り値として受け取る．多数のサーバとの同時接続が可能となっている．

(2) LDAP サーバから認証を受けるために ldap_bind() を実行する．

(3) いくつかのオペレーションを行ない結果を得る．ldap_search() といった操作要求を行ない，結果を得て，ldap_parse_result()，ldap_first_entry()，ldap_next_entry() を用いて得た結果を解析する．

(4) セッションを閉じる．ldap_unbind() を用いて接続を切る．

より細かい仕様に関しては，文献 2) に記載されているため，ここでは省略する．

☆ ISO 10646 Universal character set Transformation Format 8

3.1.1 サーバ API

現在，サーバには標準化された仕様はないため，今回実装したサーバ API は独自仕様となっている．

サーバ API は，典型的には次の 4 つのステップで使用することとした．

(1) LISTEN ポートを初期化し，ポートに接続してくるクライアントを待つために，ldap_sv_open() を呼ぶ．クライアントからの接続が行なわれると fork して子プロセスを生成し，その子プロセスがセッションハンドルを戻り値として返る．親プロセスは，接続を待ち続ける．

(2) クライアントからのメッセージを待つために，ldap_sv_required() を呼び，メッセージの到着を待つ．

(3) 到着したメッセージに従い処理を行ない，返答する．

(4) クライアントからの unbind を受け取ると，ldap_sv_close() によって接続を切る．

4. LDAP サーバ

LDAP 自動設定に用いる LDAP サーバのプロトタイプ (以後，ldapd と呼ぶ) を実装した．今回実装した ldapd は，特に LDAP 自動設定を意識した設計にはなっておらず，LDAP を用いたディレクトリの仕様の範囲内でのみ機能する．このことは，今回対象としている機能を満足する程度であれば，特殊な自動設定用サーバが不要であることを示しており，既存の LDAP サーバで構築できることを表している．以降に，ldapd の設計と構造について述べる．

なお，実装は FreeBSD 2.2.6 上で行なった．プログラムのコンパイルには，C コンパイラである gcc version 2.7.2.1 と字句解析ルーチンの生成ツールである flex version 2.5.4，構文解析ルーチンの生成ツールである GNU Bison version 1.25 を用いた．

4.1 構　　成

プロトタイプサーバの構成を図 3 に示す．

4.2 LDAP サーバ API

3.1.1 項で説明した，試作の LDAP サーバ API を使用している．ここでは，TCP/IP ソケットを用いた LDAP メッセージの送受信を行なっている．

4.3 LDAP サーバ処理実行部

サーバ API から受け取ったメッセージ (構造体) の処理内容に従い，処理を行なう．ディレクトリエントリの操作を伴う場合は，DIT 構造変換部を通じてディレクトリエントリの情報を取得する．処理結果としての返答のためのメッセージ (構造体) を生成し，サー

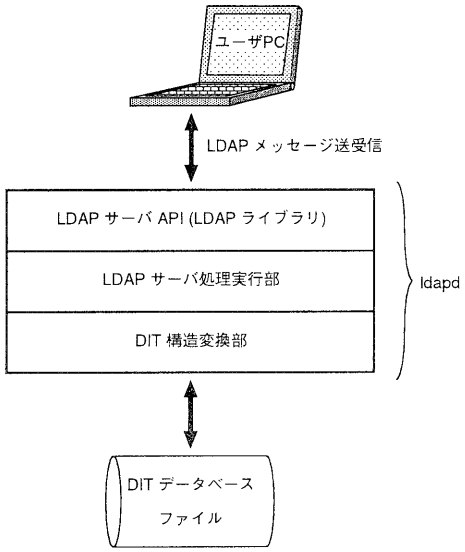

図 3 LDAP サーバの構成：ldapd は，大きく 3 つの部分から構成される．まず，LDAP メッセージとサーバ内部で使用する構造体との変換を行なう LDAP サーバ API と，search といったサーバ内の処理を行なうサーバ処理実行部，ファイルとして保持している DIT (Directory Information Tree, ディレクトリで保持されている情報木) の入出力を行なう構造変換部である．

Fig. 3 Structure of LDAP server.

バ API を通じてクライアントへ返答する．

4.4 DIT 構造変換部

ディスク上に記録してあるディレクトリエントリ (ディレクトリ内の情報の管理単位の一つ) のデータベース (以降，ディレクトリデータベースファイルと呼ぶ) から情報をディレクトリエントリとして読み出す．書き出しは未実装である．

5. LDAP 自動設定クライアント

LDAP 自動設定クライアントのプロトタイプ (以降，クライアントと呼ぶ) を実装した．本章ではクライアントの実装について述べる．

5.1 クライアントの構成

クライアントは，大きく分類して

a) LDAP メッセージを LDAP サーバとやりとりする「クライアント API 部」

b) ユーザとのインタフェイスを行う「UI 部」

c) アプリケーションプログラムの設定を変更する「設定処理部」

の 3 つの部分から構成される．

なお，クライアントの実装は Microsoft WindowsNT4.0 上の Borland C++Builder3 で行った．

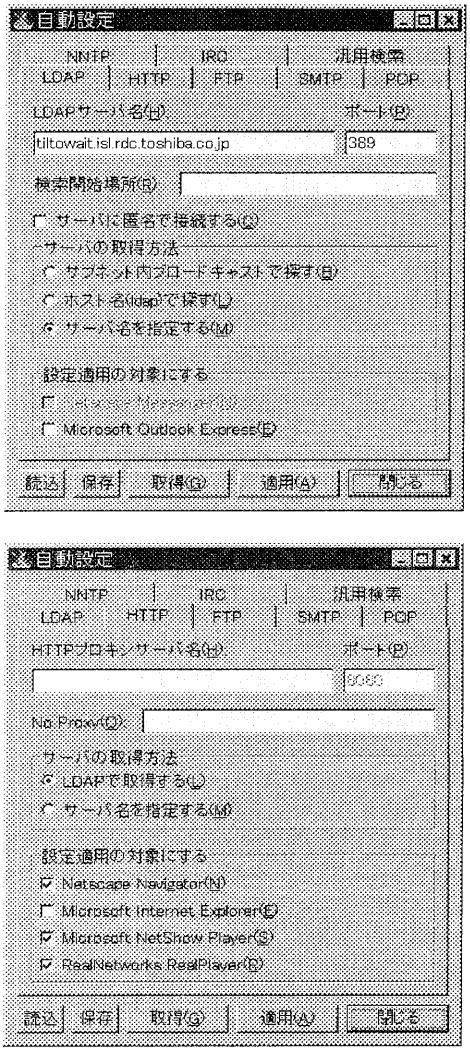

図 4 クライアントの外観：プロトコルごとの設定を各ページで行い，全体の動作を下のボタンで操作する．左図は LDAP の設定ページ，右図は HTTP の設定ページ．FTP, SMTP, POP, NNTP, IRC の各プロトコルの設定ページは右図の HTTP の設定ページとほぼ同様である．

Fig. 4 Display image of auto-configuration client.

クライアントは Microsoft Windows95 / Windows98 / WindowsNT4.0 上で動作する．

5.2 クライアント API 部

クライアント API 部は，UNIX とソースコードを共有するため C で (C++ 拡張部分は使わずに) 記述し，WinSock (Microsoft Windows 上のソケットインタフェースルーチン) を使用した．また再利用性を高めるため，このモジュールを DLL (Dynamic Link

表 2　アプリケーションプログラムの設定情報格納手法の概略

Table 2　How to setup the configuration information for commercial applications.

アプリケーション名	設定情報格納状態
Netscape Navigator/ Messenger/ Collabra	レジストリキー HKEY_CURRENT_USER\Software\Netscape\Netscape Navigator\Main\Install Directory に書かれているインストールディレクトリの \Users\${ ユーザ名 }\prefs.js というファイルの JavaScript
Microsoft Internet Explorer	レジストリキー HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrensVersion\Internet Settings の下の，ProxyEnable, ProxyServer, ProxyOverride といったエントリの値
Microsoft Outlook Express	レジストリキー HKEY_CURRENT_USER\Software\Microsoft\Internet Accoune Manager の下の，Default Mail Account, Default News Account, Default LDAP Account の値として書かれた文字列 (仮に ${ 文字列 } とする) が，サブキー \Accounts\${ 文字列 } としてあり，その中の POP3 Server, SMTP Server, NNTP Server といったエントリの値
Microsoft Internet Mail and News	レジストリキー HKEY_CURRENT_USER\Software\Microsoft\Internet Mail and News の下の，Mail はサブキー \Mail の Default POP3 Server, Default SMTP Server といったエントリの値，News はサブキー \News の DefaultServer といったエントリの値
Microsoft NetShow Player	レジストリキー HKEY_CURRENT_USER\Software\Microsoft\NetShow\Player\Local の下の，ProxyEnable, ProxyHost, ProxyPort といったエントリの値
RealNetworks RealPlayer	レジストリキー HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Software\Progressive Networks\RealAudio Player\${ バージョン番号 }\Preferences の下の，NotProxy1, HTTPProxyHost, HTTPProxyPort といったエントリの値 ただし，Version 6.0 (RealPlayerG2) 以降はレジストリキー HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Software\Real Networks\RealPlayer\${ バージョン番号 }\Preference に変更
Microsoft Chat	レジストリキー HKEY_CURRENT_USER\Software\Microsoft\Microsoft Chat の下の，IRCServerd, ServerList といったエントリの値
Chocoa	レジストリキー HKEY_CURRENT_USER\Software\Fujitsu\CHOCOA\IRC の下の，HostName, Port といったエントリの値

Library) にして本体のプログラムと分離した．

5.3 UI 部

UI 部は，設定項目が多岐にわたるため，ページコントロールを用い，プロトコルごとにページを割り振った (図 5 参照)．

LDAP のページでは，LDAP サーバの発見方法 (ldap というホスト名の付いたローカルドメインのサーバを探すか，手動で指定するか)，検索ベース，接続を匿名にするか，などを設定する．これらはこのクライアント自身の動作を決定する点で他のページとは異なる．なお，他のページと同様に，LDAP を利用するアプリケーションソフトウェアに対して，これらのデータを適用するかどうかも選択できる．

HTTP, FTP, SMTP, POP, NNTP, IRC のそれぞれのプロトコルのページは，サーバの取得方法 (LDAP で取得するか，手動で指定するか)，および設定適用の対象 (どのアプリケーションソフトウェアに設定を反映するか) を指定する．サーバの取得方法を手動にした場合は，サーバ名，ポート番号など，そのプロトコルに対応する情報を入力しておく．これらの情報はレジストリに記録され，次回起動時には前回の設定が再現されるようにした．

ページの下には「読込」,「保存」,「取得」,「適用」,「閉じる」の各ボタンを配置した．プログラム起動後「取得」を押すことで，LDAP サーバに接続し情報を取得する．取得結果は各ページに表示される．その後，「適用」を押すと，取得したデータを設定対象に指定したプログラムに適用する.「保存」は，自動取得した情報をファイルとして保存しておく補助機能,「読込」は，ファイルに保存した情報を読み込む補助機能である．必要な操作が終わったら「閉じる」でプログラムを閉じる．

通常はプログラムを起動後,「取得」「適用」「閉じる」の 3 ステップで設定が完了する．なお，起動時オプション等でこの一連の操作を自動化することでさらに手間を省くことは容易である ☆.

5.4 設定処理部

設定処理部は，主にアプリケーションソフトウェアの設定情報が記録されている設定ファイルおよびレジストリを操作する．この部分はアプリケーションプログラムによってまちまちであるため，個別の対応が必要になる．対応したアプリケーションソフトウェアは，次の 10 種類である．

- Netscape Navigator
- Netscape Messenger
- Netscape Collabra

☆ 未実装

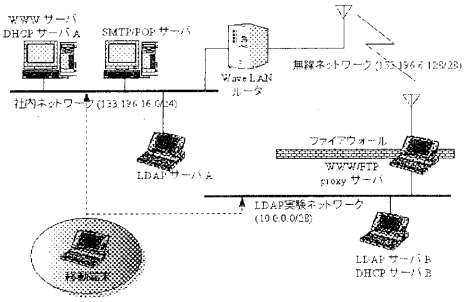

図 5 動作実験ネットワークの構成：2 つのネットワークにそれぞれ LDAP サーバを設置する．ネットワーク間を移動するクライアント PC は，LDAP サーバにアクセスし，接続された先のネットワーク環境に設定変更することで各種ネットワークサービスを利用することができる．

Fig. 5 Network configuration of the experimental system.

- Microsoft Internet Explorer
- Microsoft Internet Mail & News
- Microsoft Outlook Express
- Microsoft Chat
- Microsoft NetShow Player
- RealNetworks RealPlayer
- Fujitsu Chocoa

ここでは，それぞれのアプリケーションプログラムの設定情報格納手法の解析結果の概略を表 5.1 に示す．

6. 動作結果

実装したサーバおよびクライアントを図 5 に示すような異なるセグメントからなるネットワーク上にそれぞれ設置し，動作確認を行った．

移動端末に見立てたノート型 PC 上でクライアントを動作させ，接続するネットワークを移動する度に LDAP サーバからの情報を利用してアプリケーションソフトウェアの設定を変更した．変更後，それぞれのネットワーク上のサービス (WWW, FTP, SMTP など) が 3 ステップの操作のみで正しく受けられることを確認した．

7. おわりに

本稿では，LDAP を用いたアプリケーションプログラムの自動設定ツールとして，2.2 節に挙げた機能のうち，

(機能 1) アプリケーションプログラムの設定に必要な情報の配送と取得

(機能 2) ユーザ PC のアプリケーション設定への自動反映

の 2 つの機能の設計と試作について述べた．実験ネットワークにおいて，本ツールにより 3 ステップで設定が完了できることを確認した．しかし，今回の試作では，ネットワークサービスを対象とした設定の自動化と，ユーザ PC 側で設定情報の取得方法を知っているという方針での試作となっている．従って，上記の 2 機能において，

(残された課題 1) ネットワーク以外の設定情報の扱い

(残された課題 2) 取得すべき設定情報の問い合わせ方法の動的な適応

という課題が残されている．また，今回の実装のように，アプリケーションの設定を LDAP 自動設定クライアントに一任する形式においては，

(残された課題 3) 設定変更に必要な処理内容の動的な適応

(残された課題 4) アプリケーション設定の解析の必要性

という課題も残されている．

更に，未実装の機能の実装に加え，新たなアプリケーションの自動設定においても，プログラムコードを変更せずに済むよう適応力を持たせるために，アプリケーションの設定情報だけでなく，LDAP 自動設定クライアントの動作 (処理内容) に関する情報も LDAP サーバに登録するようにする方式の検討が必要である．

参考文献

1) RFC2251 - RFC2256, Lightweight Directory Access Protocol (v3) (Dec. 1997)
2) draft-ietf-ldapext-ldap-c-api-01.txt, The C LDAP Application Program Interface (7 Aug. 1998)
3) RFC2244, ACAP – Application Configuration Access Protocol (Nov. 1997)
4) draft-ietf-lsd-client-finding-00.txt, LDAP Clients Finding LDAP Servers (Jan. 1998)
5) draft-ietf-svrloc-protocol-v2-10.txt Service Location Protocol, Version 2 (9 Oct. 1998)
6) draft-ietf-svrloc-api-06.txt, An API for Service Location (31 Jul. 1998)
7) RFC2131, Dynamic Host Configration Protocol (Mar. 1997)