ネットワークソフトウエアの構成法に関する研究

93b527 加田優子 93b536 加藤さおり
93b574 村山真章

指導教員　野呂昌満

1　はじめに

ソフトウェアの部品化および部品の再利用を目的として、参照アーキテクチャ [2] やフレームワーク [4]、デザインパターン [1] などの研究が行われている．部品を再利用することによって、ソフトウェア開発における時間や労力を削減することが期待できる．

現在、ネットワークソフトウェアの多くは、 UNIX のシステムコール等を用いて、手続き指向プログラミング言語 C によって実現されている．したがってネットワークソフトウェアは、オブジェクト指向による部品化と参照アーキテクチャが確立されていない． 本研究で対象とするネットワークソフトウェアとは、 IP プロトコルにもとづいて複数のプロセスが異なるノード上で協調して動作する、UNIX の ftp、 telnet 等のソフトウェアのことである．

本研究の目的は、ネットワークソフトウェアの開発におけるネットワークソフトウェアの参照アーキテクチャとそのフレームワークを定義することである．

一般にソフトウェアの部品化を行う際には、以下のような問題点がある．

- 部品の定義が曖昧なので、部品化が困難である
- 部品は要求に依存しているので、異なる要求に対しては再利用できない

我々は、フレームワークを部品として作成し、部品の使用法となる参照アーキテクチャをネットワークソフトウェアの作成者に提供することで、以上の問題点を解決できると考えた．フレームワークは、応用領域 (応用ソフトウェアの範囲) を限定することによって得られる、汎用性は低いが、粒度が大きく、再利用率が高い部品である．

我々は、ネットワークソフトウェアの参照アーキテクチャとしてネットワーク三層モデルを考案した．それにもとづいて、 UNIX コマンド (ftp、 telnet 等) をオブジェクト指向プログラミング言語である C++ を用いて再設計・実現した． 実現したプログラムの中から共通部分を抽出後、ネットワークソフトウェアにおけるフレームワークを設計した．

2　ネットワークソフトウエア概観

通信形態を分類する次元

UNIX 上の既存のネットワークソフトウェアの通信形態は、以下の次元で分類できる．

- プロトコル
  - TCP
  - UDP
- 通信方式
  - 同期通信
  - 非同期通信
- ドメイン
  - UNIX ドメイン
  - INET ドメイン

クライアント / サーバ モデル

クライアント / サーバモデルは、サービスを要求するクライアントと、サービスを提供するサーバに役割を分割したモデルである．

クライアント / サーバモデルは、分散環境において、サーバを一つのホストだけで起動し、そのサーバに全てのクライアントがサービスを要求することによって、他のホストでサーバを起動させる必要がなくなる．この利点から、クライアント / サーバモデルは、ネットワークソフトウェアの参照アーキテクチャとして多く用いられている．

ネットワークソフトウエア作成における問題点と解決策

現状におけるネットワークソフトウェアの作成には、以下の問題点がある．

- 通信に関して詳細な知識が必要
- 通信に関する細かいレベルまでプログラミングが必要

これらの問題点の解決策として、ネットワークソフトウェアにおける参照アーキテクチャとフレームワークを定義する．その手段として、既存のネットワークソフトウェアをオブジェクト指向の考えにもとづいてクライアントとサーバをオブジェクトとして再設計・実現する．オブジェクト交換形式である ORB の機構を導入し、ネットワークソフトウェアにおける通信と通信以外の処理を分離して扱う．

3　参照アーキテクチャとフレームワーク

参照アーキテクチャ

参照アーキテクチャとは、部品を応用ソフトウェアのどこに配置すべきかを示すもので、ソフトウェアのおおまかな構成を規定したものである．したがって、応用ソフトウェアの作成者は、参照アーキテクチャを参考にして、応用ソフトウェアを作成することができる．

参照アーキテクチャの例として、 Donovan の三層モデル [2]、 OSI 参照モデル、クライアント / サーバモデルがあげられる．

フレームワーク

フレームワークとは、特定の応用領域における互いに関係のある部品の集合である． 以下の理由により、フレーム

ワークを作成するには、クラスの機構を提供するオブジェクト指向言語が適している.

- 部品をクラスとして実現できる
- 部品間の関係をクラス階層として表現できる

すなわちフレームワークは、抽象クラスと具象クラスを部品としたクラス階層として実現できる. フレームワークにおいて、使用者コードを付け加える部分をホットスポット、使用者コードを付け加えることができない部分をフローズンスポットと呼ぶ.

フレームワークを利用して特定の応用領域における応用ソフトウェアを作成すると以下の利点がある.

- フレームワークの部品は汎用のライブラリの部品に比べると再利用率が高い
  フレームワークは応用領域を限定して作成される部品なので、部品の粒度は大きくなる. したがって、汎用の部品に比べると再利用率が高くなる.
- コードだけでなく応用ソフトウェアの設計も再利用できる
  フレームワークの利用者は、ホットスポットを使用者コードで補うことによって、応用ソフトウェアを作成することができる. フローズンスポットは、新たに設計し作成する必要がないので、応用ソフトウェアの設計も再利用することができる.

フレームワークと参照アーキテクチャの関係

フレームワークと参照アーキテクチャの関係は、”部品と部品の使用法”の関係として考えることができる. フレームワーク自体をひとつの部品として考えると、参照アーキテクチャの各層は、適用する部品とその配置箇所を示すので、参照アーキテクチャは部品の使用法と考えることができる (図 1参照).

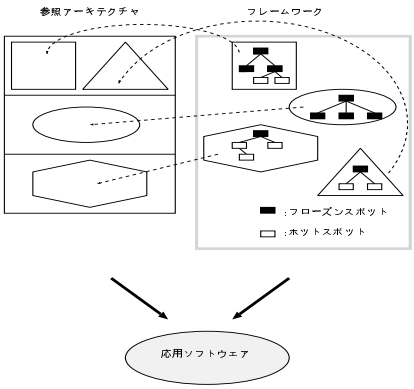

図 1: 参照アーキテクチャとフレームワークの関係図

4 既存のネットワークソフトウェア

ftp

TCP を利用した、ネットワーク上の端末間でファイル転送を実現する標準プロトコルの 1 つである. ftp には、ローカルホストからリモートホストへファイル転送をするアップロードと、リモートホストからローカルホストへファイル転送をするダウンロードの 2 種類のサービスがある. データを送信するさいは、ネットワーク標準の文字コードやデータフォーマットに変換してからファイルを転送する. データを受信するさいは、転送されたファイルを使用者の端末特性に合わせて変換する.

telnet

TCP を利用した仮想端末機能を実現する標準プロトコルの 1 つである. 仮想端末機能とは、ローカルホストからネットワークを介してリモートホストを利用する機能である.

talk

UDP を利用したリアルタイムで画面上に文字を表示して 1 対 1 の会話を実現するネットワークソフトウェアである. 非同期通信方式を採用している.

5 ORB(Object Request Broker)

2 章で述べたネットワークソフトウェア作成の際における問題点を解決するために、 ORB を使用して、階層構造を整理する. ORB(Object Request Broker) とは、分散環境におけるオブジェクト間のメッセージのやり取りをネットワークを意識せずに通信させるための機構である.

ORB を用いた代表的なアーキテクチャとして CORBA (Common Object Request Broker Architecture) が挙げられる. CORBA は、分散したオブジェクト間のプロトコルを定義し、分散したオブジェクトがどこに存在するのかを使用者に意識させないアーキテクチャである.

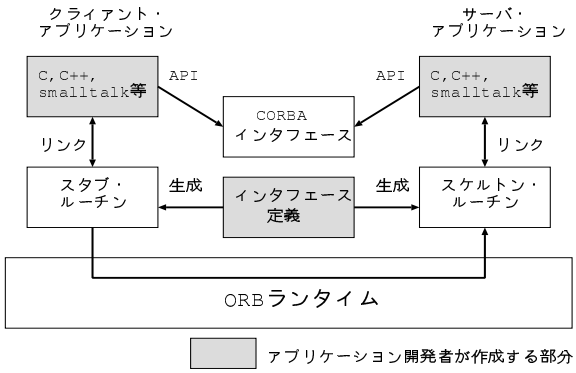

図 2: CORBA アプリケーションの構造

例として、 ORB の仕様である CORBA のアーキテクチャの構造を図 2に示す. 分散しているオブジェクトが提供するクライアントやサーバのインタフェースにおいて、それぞれが異なる言語で記述されているとお互いに通信ができない. アプリケーション作成者は、共通したインタフェース定義言語 (IDL) でインタフェースを記述する. このIDL 定義を専用コンパイラ (IDL コンパイラ) でコンパイルすることにより、クライアントとサーバの通信を仲介するクライアント側のスタブとサーバ側のスケルトンが自動生成される. スタブとスケルトンは、クライアント側とサーバ側それぞれの応用ソフトウェアで使用するプログラミング言語で記述されている. このスタブとスケルトンを使用することにより、 ORB の役割を果たす.

6 ネットワーク三層モデルとライブラリの実現

我々は、ネットワークソフトウェアの参照アーキテクチャを考案し、ネットワーク三層モデルと名付けた. ネットワーク三層モデルは、

- アプリケーション層
- ORB 層
- 通信層

の 3 つの層から構成される. ネットワーク三層モデルの各層における部品を作成することによって、ネットワークソフトウェアの部品化を行うことができる. 本研究では、オブジェクト指向の考えにもとづいてネットワークソフトウェアの部品化を行うので、各層における部品をオブジェクトとして実現した. 図 3に、ネットワーク三層モデルを示す.

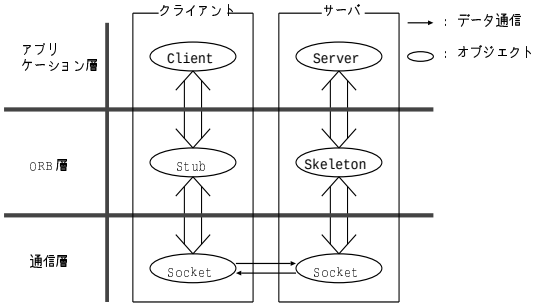

図 3: ネットワーク三層モデル

6.1 アプリケーション層

アプリケーション層におけるオブジェクトの役割は、ネットワークソフトウェアの通信以外の処理を行うことである. クライアント側のオブジェクトを Client オブジェクト、サーバ側のオブジェクトを Server オブジェクトと呼ぶ.

アプリケーション層で行なわれる処理は、クライアント側とサーバ側では以下のように異なる.

クライアント側

- 使用者からのコマンドを解析し、後述の Stub オブジェクトにメッセージを送る
- サーバからのサービスの結果を標準出力に出力したり、ファイルに保存する等の処理を行なう

サーバ側

- 後述の Skeleton オブジェクトからのメッセージに対して使用者認証やファイル転送等のサービスを実行する

6.2 ORB 層

ORB 層におけるオブジェクトの役割は、アプリケーション層のオブジェクトが、ネットワークを意識せずにメッセージを送ることができるように仲介することである. クライアント側のオブジェクトを Stub オブジェクト、サーバ側のオブジェクトを Skeleton オブジェクトと呼ぶ.

本研究では、 UNIX 上で起動するネットワークソフトウェアを対象としているので、プロセス間通信を行う際に、 UNIX のシステムコールを利用してソケットを作成した. ソケットを用いて送受信するデータの型は文字列型でなければならない. 文字列型以外の型のデータを送信する際には、そのデータを文字列型のデータに変換（エンコード）する必要がある. データを受信する際には、文字列型のデータをエンコードされる前の型のデータに変換（デコード）する必要がある.

データのエンコードとデコードは、 ORB 層のオブジェクトによって行われる. アプリケーション層における部品の作成者は、データのエンコード、デコードを考慮する必要がない. その結果、 Client オブジェクトが Stub オブジェクトにメッセージを送ることと、 Client オブジェクトが Server オブジェクトにメッセージを送ることは同じであると考えて、 Client オブジェクトと Server オブジェクトを作成することができる.

6.3 通信層

通信層におけるオブジェクトの役割は、プロセス間でデータ通信をすることである. 通信層におけるオブジェクトを Socket オブジェクトと呼ぶ. 6.2 節でも述べたように、本研究では、 UNIX のシステムコールを用いてソケットを作成し、ソケットを用いたデータ通信をする.

6.4 ライブラリの実現

我々は、ネットワーク三層モデルをもとに、 ftp、 telnet、 talk のオブジェクト指向の考えにもとづいて、再設計・実現を行なった. 実現のさいには、オブジェクト指向プログラミング言語である C++ を用いた. Stub オブジェクト、 Skeleton オブジェクト、 Socket オブジェクトの実現について述べる.

ORB層

Stubオブジェクトの実現

Stubオブジェクトは以下のメンバを持つように設計・実現した.

- Socketオブジェクトへのポインタ
- コンストラクタ

  サーバのIPアドレスとポート番号を引数にとり、ソケットを作成する.
- メンバ関数 SendMessage()

  メソッド名とその引数をエンコードして、Skeletonオブジェクトに送信する. また、メソッドの戻り値をデコードしてClientオブジェクトに返す.
- メンバ関数 ConnectClose()

  ソケットを閉じる.

Skeletonオブジェクトの実現

Skeletonオブジェクトは以下のメンバを持つように設計・実現した.

- Socketオブジェクトへのポインタ
- Serverオブジェクトのメソッドへのポインタを要素として持つリストへのポインタ
- コンストラクタ

  サーバのIPアドレスとポート番号を引数にとり、ソケットを作成する.
- メンバ関数 HandleMessage()

  Serverオブジェクトのメソッドへのポインタをリストに登録する
- メンバ関数 MakeChild()

  サーバの子プロセスを生成する. 親プロセスでは、子プロセスのソケットを閉じて、新たな接続の要求を待つ. 子プロセスでは、
  - 親プロセスのソケットを閉じる
  - クライアントからのメッセージをデコードし、Serverオブジェクトにメッセージを送る
  - 戻り値をエンコードしてStubオブジェクトに送信する
  - 子プロセスのソケットを閉じる

  ことを行なう.

通信層

Socketオブジェクトの実現

UNIXのシステムコールを使用するので、利用できるソケットは、

- ドメイン（UNIXドメイン・ INETドメイン）
- プロトコル（TCP・ UDP）
- クライアント/サーバ

の組合せにより8種類考えられる. 我々は、多重継承を用いて8種類のクラスSocketを作成した. 図4に、クラスSocketの例を示す.

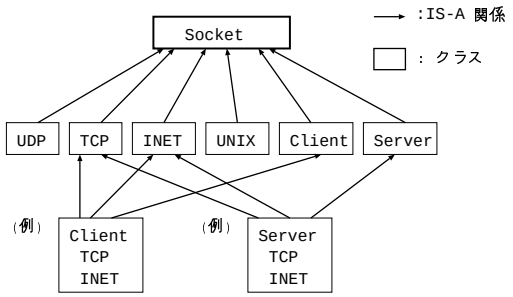

図 4: クラス Socket

7 UNIXコマンドの再設計と実現

7.1 ftpの再設計と実現

我々は図5に示すようにftpをオブジェクト指向の考えにもとづいて、再設計・実現した. ftp-Userオブジェクトは、使用者によって入力されたコマンドを解析し、ftp-Clientオブジェクトにメッセージを送る. Fileオブジェクトは、以下のファイルに関する操作を行なう.

- 読み込み・書き込みでファイルを開く
- ファイルに文字列を保存する
- ファイルから文字列を取り出す
- ファイルのバイト数を調べる
- ファイルを閉じる

ftp-Serverオブジェクトとftp-Clientオブジェクトは、Fileオブジェクトにメッセージを送ることによって、ファイルに関する操作を行なうように実現した.

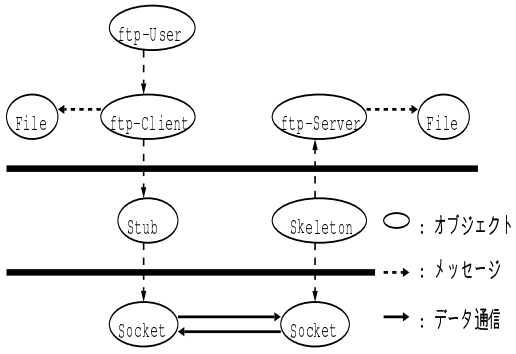

図 5: ネットワーク三層モデルにもとづいたftp

7.2 telnetの再設計と実現

我々は図6に示すようにtelnetをオブジェクト指向の考えにもとづいて、再設計・実現した. telnet-Userオブジェクトは、使用者によって入力されたコマンドを解

析し、 telnet-Clientオブジェクトにメッセージを送る. Shellオブジェクトは、 UNIXのシェルを起動してクライアントから送られてきたコマンドを実行する.telnet-Serverオブジェクトは、 Shellオブジェクトにメッセージを送ることによって、コマンドを実行し、実行結果をクライアントに返す.telnet-Clientオブジェクトは、実行結果を標準出力に出力する.

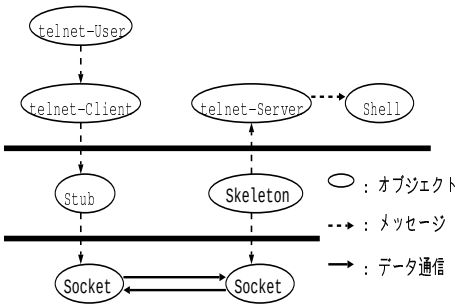

図 6: ネットワーク三層モデルにもとづいた telnet

7.3 talk の再設計と実現

我々は図 7に示すように talk をオブジェクト指向の考えにもとづいて、再設計・実現した. talk-User オブジェクトは、使用者によって入力された文字を読み込み、通信相手に送信する文字か終了コマンドかを解析する.Conversationオブジェクトは、以下の端末に関する操作を行なう.

- 端末の属性の変更
- 使用者が入力した文字を画面に表示する

talk-Serverオブジェクトと talk-Clientオブジェクトは、Conversationオブジェクトにメッセージを送ることによって、クライアントとサーバの端末の画面に文字を出力させる.

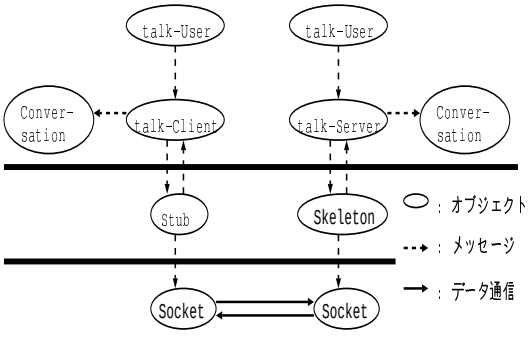

図 7: ネットワーク三層モデルにもとづいた talk

8 ネットワークソフトウエアのフレームワークの設計

我々は、 7章でオブジェクト指向の考えにもとづいて再設計したコードをもとに、ネットワーク三層モデルのアプリケーション層におけるフレームワークを設計した. フレームワークを定義するさいの手順を以下に示す.

1. ftp、 telnet、 talk の Client オブジェクトと Server オブジェクトに共通する部分をそれぞれ上位クラスとして抽出する
2. 1 を繰り返すことによってクラス階層を定義し、フレームワークを定義する

使用者認証に関するフレームワークの設計

ネットワークソフトウェアの中には、 ftp や telnet のように使用者の認証を行うものがある.

クライアント側とサーバ側に使用者認証オブジェクトが存在する. クライアント側の使用者認証オブジェクトは、使用者から使用者名とパスワードの入力をうながし、サーバ側へ送信する. サーバ側の使用者認証オブジェクトは、受信した使用者名とパスワードをもとに使用者認証を行い、その結果をクライアント側に返す.

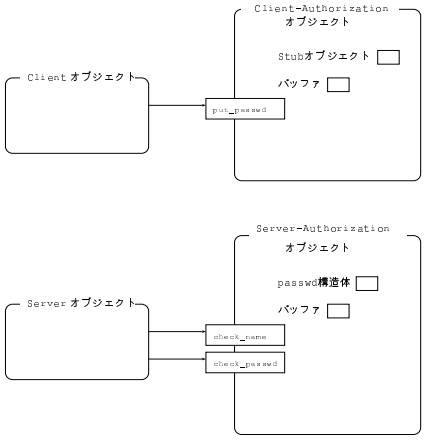

図 8: 使用者認証オブジェクト

状態遷移に関するフレームワークの設計

クライアントとサーバには状態の遷移があるので、クライアントとサーバの状態を管理する状態遷移表オブジェクトを設計し、状態遷移に関するフレームワークとして設計した.

クライアント側とサーバ側に状態遷移表オブジェクトが存在する. 状態遷移表オブジェクトは、クライアントとサーバの状態を管理する.

クライアントとサーバの状態は、以下のように遷移する.

- クライアントの状態は使用者からのコマンド入力によって遷移する
- サーバの状態はメッセージの受理によって遷移する

使用者が入力するコマンドは、 Client オブジェクトのメソッドであり、サーバが実行するサービスは Server オブジェクトのメソッドである. クライアントが要求できるコマンドはクライアントの状態に依存し、サーバが実行できるサービスはサーバの状態に依存するので、状態遷移表オブジェクトが以下のデータメンバを持つと考えた.

- クライアント (サーバ) の現在の状態
- クライアント (サーバ) の状態遷移表

状態遷移表オブジェクトは、以下のメソッドを持つ.

- クライアント (サーバ) の状態遷移表を作成する
- Client オブジェクト (Server オブジェクト) のメソッド名を引数にとり、そのメソッドが実行できるかどうかを調べる
- クライアント (サーバ) の現在の状態を遷移させる

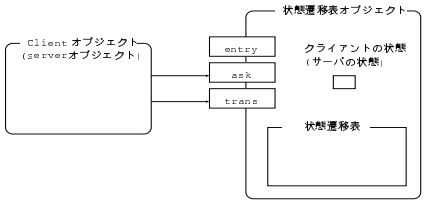

図 9: 状態遷移表オブジェクト

9 おわりに

我々はネットワークソフトウェアの参照アーキテクチャであるネットワーク三層モデルを考案し、それにもとづいて UNIX コマンド (ftp、 telnet、 talk) をオブジェクト指向プログラミング言語である C++ を用いて再設計・実現した. この結果、 UNIX コマンド (ftp、 telnet、 talk) はネットワーク三層モデルによって、整理できることが確認できた. ネットワークソフトウェアのフレームワークとして、使用者認証、状態遷移、コマンド解析の 3 つを定義できることが分かり、使用者認証オブジェクト、状態遷移表オブジェクトの設計を行なった.

状態遷移表オブジェクトとコマンド解析オブジェクトを設計するさいに、デザインパターンのステイトパターンとコマンドパターンを利用できる. ステイトパターンとコマンドパターンを組み合わせることによって、状態遷移とコマンド解析に関するフレームワークを設計し実現することと使用者認証に関するフレームワークを実現することを来年度の課題とする.

謝辞

本研究を進めるにあたり、御指導して下さった指導教員の野呂昌満助教授、大学院生の塩田康隆氏、山野篤氏、平野純二氏をはじめとする皆様に心から感謝致します。

参 考 文 献

[1] Erich Gamma,John Vlissides, Ralph Johnson, Richard Helm オブジェクト指向における再利用のためのデザインパターン, ソフトバンク,1995.

[2] John J.Donovan,*Business Re-engineering with Information Technology*, P T R Prentice Hall,1994.

[3] Ron Zahavi, Thomas J. Mowbray*The Essential CORBA*, John Wiley &Sons,Inc,1995.

[4] Ted Lewis 他 著 *OBJECT ORIENTED APPLICATION FRAMEWORKS*, Manning Public Co.,1995.