ネットワークソフトウェアの構成法に関する研究

96B506 青木 俊介

指導教員　野呂 昌満

1　はじめに

既存のネットワークソフトウェアの多くは, 構造が整理されておらず, 理解するのが困難である. また, UNIX 上でネットワークソフトウェアを作成する場合, 再利用されている部品がシステムコールライブラリという粒度の低いものであるという問題点が挙げられる.

この問題を解決するために, 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 IParch[1][2] が提案された. また, その応用フレームワークである IPaf が設計, 実現された. しかし, 提案された IParch, IPaf には,

- 時間切れ処理が汎用的な処理として実現されていない.
- 既存のネットワークソフトウェアとの互換性が考慮されていない.

という問題点がある. 本研究では, これらの問題を解決するための改版したアーキテクチャ IParch-R を提案し, そのフレームワークである IPaf-R を実現した.

2　改版したアーキテクチャの提案

2.1　時間切れ (time out) 処理

ネットワークソフトウェアでは通信障害などによる時間切れを処理しなければならないが, 時間切れのような例外的な処理はソフトウェアの構造を乱す要因の一つである. アーキテクチャでは, ソフトウェアの構造を整理したまま時間切れ処理を扱えなければならない. 我々は, 時間切れ処理を IParch-R の通信層に追加した.

IParch では, 通信障害, 相手側の故障による時間切れ処理をプロキシ層で実現していた. プロキシ層で実現する場合は, `Timer` オブジェクトをプロキシ層に配置する. 時間切れがおきると `Timer` オブジェクトが行動処理オブジェクトを生成する. 行動処理オブジェクトを状態遷移層に渡すことで, 時間切れ処理を実行する.

しかし, 通信障害, 相手側の故障による時間切れ処理は, ネットワークソフトウェアを作成するさいにアプリケーションに依存しない汎用的な処理である. IParch-R の通信層は, アプリケーションに依存しない汎用的な部品をあつかう層なので, 通信層で実現することにした. 時間切れが起こった場合には, 通信層で時間切れを表す特別なメッセージを生成し, 上位の層ではサーバからのメッセージと同様にしてあつかう.

2.2　既存のネットワークソフトウェアとの互換性

IParch では, 独自の方法でプロトコルが実現されていたので, 既存のネットワークソフトウェアとの互換性が保証されていなかった. IParch-R では, プロキシ層で既存のネットワークソフトウェアとの互換性を保証する. プロキシ層で互換性を保証する利点は, サーバソフトウェアの実現方法が変更されても, プロキシ層のオブジェクトを変更するだけで対応できることである.

図 1に IParch-R の概略を示す.

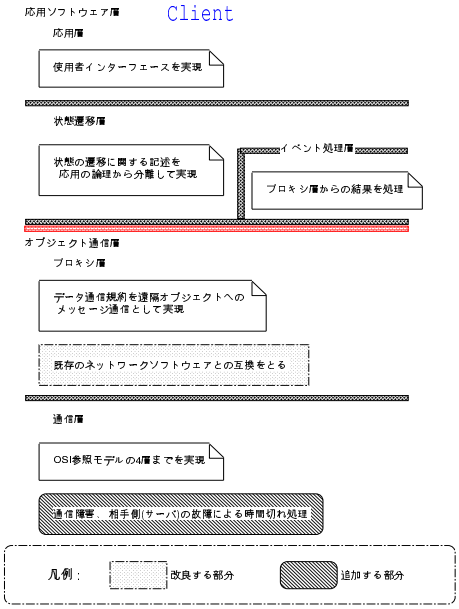

図 1: IParch-R の概略図

3　スタブの自動生成

我々は, プロキシ層のオブジェクトであるスタブを自動生成する. スタブを自動生成する利点は,

- 通信に関するコードの実現が省略できる
- メッセージ通信を遠隔オブジェクトへのメソッド呼び出しにできる.

ことである.

IParch では, サーバ / クライアントともに作成した場合に, サーバ応用層のクラスのインターフェースからスタブが自動生成されていた. IParch は, 構造の整理を目的として作成されているので, サーバソフトウェアを解析し, スタブを自動生成するのは容易である. しかし, 既存のネットワークソフトウェアは, 構造が整理されていないので, スタブを自動生成するのは困難である. 我々は, 既存のネットワークソフトウェアの通信に必要なプロトコルを

表す仕様を作成し, スタブを自動生成する. 仕様を作成するために, 既存のネットワークソフトウェアを調査し, 以下のことが分かった.

- 通信におけるデータは, 文字列である.
- コマンドは, コマンド名とコマンドに付随した引数で構成される.
- プロトコルによって, データの終端を表す方法が異なる. POP の例では, 終端条件は "\r\n.\r\n" である.
- データの転送の時に置換される文字列がある. POP の例では, 文中の "\n." を "\n.." にすることである.

調査した結果から, スタブを自動生成する仕様には,
(1) コマンド名とその引数
(2) 終端条件
(3) 置換文字列
が必要である.

図 2に仕様に基づいた記述を示す. "format" は (1), "Eod" は (2), "replace" は (3) を表す.

```
class Server {
    Res *user(char *user) {
        char format[] = "USER %s\n";
    };
    Res *retr(int num) {
        char format[] = "RETR %d\n";
        char Eod[] = "\r\n.\r\n ";
        char *replace[2] ={"\n..","\n."};
    };
};
```

図 2: 自動生成系に渡される記述 (例:POP)

4 改版したフレームワークの設計, 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である IParch-R をもとに応用フレームワーク IPaf-R を設計, 実現した.

IParch-R の構成に基づいて, 応用層に Command_Table クラス, 状態遷移層に State_Trans クラス, イベント処理層に Process_Event クラス, プロキシ層に Stub クラス, 通信層に Socket クラスを作成した. また, それぞれのクラス間で受け渡されるクラスに, Command クラス, Action クラス, Res クラスを作成した.

サーバからの結果が正常な場合とエラーの場合で処理がことなるので, Action に 3 つのメソッドを追加した.

- 正常な場合に呼ばれる execute メソッド.
- サーバーのエラーの場合に呼ばれる error メソッド.
- 時間切れの場合に呼ばれる timeout メソッド.

3 つのメソッドを追加することで, 正常な結果が返された場合の処理と, エラーが返された場合の処理を分けることができた. 以下に, サーバへメッセージを送った後の実行の概略を示す.

1. サーバからの結果が Socket から Stub を通じて Process_Event クラスに渡される.
2. サーバからの応答がなく時間切れした場合, Socket から Stub を通じて Process_Event クラスに時間切れを通知する.
3. Process_Event クラスは, 正常な結果かエラーかを判断し, 対応するイベントを生成する.
4. 生成されたイベントに対応する Action のメソッドが実行され, コマンドの処理が終了する.

我々は, IPaf-R を用いて POP と HTTP のクライアントを作成した. 2 つのネットワークソフトウェアを作成したことにより

1. Socket, State_Trans は, フローズンスポットとなる.
2. IPaf-R を用いてクライアントネットワークソフトウェアを作成した場合, コマンドの追加, 変更, 応用層の実現の変更が容易になる.

ということを確認した.

5 おわりに

我々は, クライアントのアーキテクチャを提案し, フレームワークを作成した. 作成したフレームワークをもとに,POP,HTTP のクライアントを試作した. 実現にあたってはオブジェクト指向言語の C++ を用いた. また, スタブ自動生成系の実現にあたっては,flex,bison++ を用いた.

今後の課題として

- サーバのアーキテクチャの提案
- 他のネットワークソフトウェアを調査, 試作

があげられる.

参 考 文 献

[1] Masami Noro,Kunio Goto, "An Architecture and a Framework for IP Applications", , *Proceedings of APSEC'97*, pp. 191–199, 1997.

[2] 森 忠夫 : ネットワークソフトウェアの構成法に関する研究, 南山大学経営学研究科修士論文, 1998.