TCP/IP のアスペクト指向実現に関する考察

M2004MM026 小久保 佳将
指導教員　野呂 昌満

1　はじめに

現存の TCP/IP 処理ソフトウェア [6] は，事実上標準仕様である RFC[4] に基づいて，安直に手続き指向で実現したものである．柔軟性や再利用の観点からオブジェクト指向技術による構造整理が試みられている [1][3][5]．しかし，コンサーン横断問題 [2] に対しては、依然部分的な解を示すにとどまっている．

我々は，組み込みソフトウェアのアスペクト指向ソフトウェアアーキテクチャスタイル (E-AOSAS) を提案している．E-AOSAS では，組み込みソフトウェアのアーキテクチャを並行に動作する状態遷移機械の集合と規定している．

本研究の目的は，TCP/IP 処理ソフトウェアを組み込みソフトウェアの典型と位置づけ，アスペクト指向技術を用いてアーキテクチャの構築を行うことである．さらに，構造を一般化して E-AOSAS の改版も目指す．

TCP/IP 処理ソフトウェアの仕様の分析，E-AOSAS の観点から，横断コンサーンの分離を行った．E-AOSAS に，実時間処理，耐故障性，セキュリティ，コンフィギュレーションコントロールアスペクトを付加した E-AOSAS+ の提案を行い，その妥当性を検討した．

2　関連研究

TCP/IP 処理ソフトウェアの構造整理を目的とした研究として，Sockets++[1]，Conduit+[3] や Patterns for Protocol System Architecture[5] がある．

Sockets++ は，UNIX ソケットのオブジェクト指向インタフェースを提供するものである．UNIX ソケットをライブラリとして提供し，再利用を図っている．Sockets++ の実現では，横断コンサーンの取り扱いについては言及されていない．

Conduit+ では，デザインパターンを利用してプロトコル処理ソフトウェアにおける構造の定義を行っている．プロトコル処理ソフトウェアのパターン利用したアーキテクチャを示している．

Patterns for Protocol System Architecture では，Conduit+ の考え方に基づいたフレームワークの構築を行っている．プロトコル処理ソフトウェアの共通部品を，仕様，設計，およびに実現の観点から分析することで，共通の部品を導出している．

これらの研究では，デザインパターンを使ってコンサーンの分離を試みている．状態遷移とアプリケーションロジックを State パターン，Command パターンを用いて分離し，構成管理は Prototype パターンで分離をしている．我々は，系統的な分析を行い，アスペクト指向技術を用いてコンサーンの分離を行った．

3　組み込みソフトウェアのアスペクト指向ソフトウェアアーキテクチャスタイル

E-AOSAS では，組み込みソフトウェアを，複数の状態遷移機械が互いにメッセージを送り，協調動作するものと規定している．状態遷移機械には待機状態と活性状態がある．システム起動時には状態遷移機械は待機状態であり，受理できるイベントを受け取った場合，活性状態へ遷移する．活性状態となった状態遷移機械は受け取ったイベントに対する処理を行い，再び待機状態に戻る．

並行状態遷移機械は，並行処理，状態遷移，およびアプリケーションロジックで構成される (図 1 参照)．並行処理を主要コンサーンとして，状態遷移，アプリケーションロジックを二次コンサーンとして分離する．

プラットフォーム独立の観点から，並行処理コンサーンを分離する．並行処理の実現は，汎用性が高く原始的な signal-wait 方式に基づく．並行処理を主要コンサーンとすることで，アスペクトとして状態遷移を分離できる．

部品化の観点から状態遷移機械とアプリケーションロジックを分離する．再利用の観点からすれば，アスペクト間記述が部品の文脈を吸収している．アスペクト間記述が部品の文脈と意味の隔りを埋める．これにより実行の文脈と部品の意味の独立が図られる．

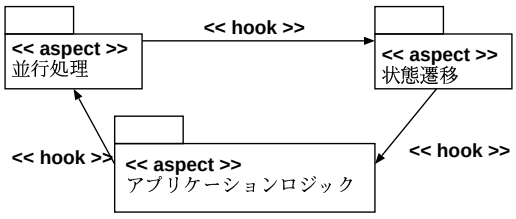

図 1　構成要素間の関係

4 TCP/IP 処理ソフトウェアのアスペクト指向アーキテクチャ

非機能特性に関する処理がコンサーンを規定するものと考え，非機能特性に関する処理をアスペクトとしてモジュール化する．仕様である RFC を分析，ならびに E-AOSAS で設計し，コンサーンの分離を試みる．

4.1 RFC における非機能特性の仕様定義

RFC では，TCP/IP 処理ソフトウェアにおける非機能特性に関する処理を下記のように規定している．

並行性

リモートホストと通信を行うために，送信処理，受信処理を並行動作させる必要がある．

実時間性

パケットの遅延などに対処するための再送タイマやコネクション維持・管理のためのタイマが必要であり，これらは仕様の中で用いられることが規定されている．

セキュリティ

暗号化処理は，各階層やコネクションの形態などの視点から考察されている．

例外処理

パケットの破損，遅延に対応するための処理を通常の通信処理と区別する．また，時間切れ時の再送処理も通常の通信処理と区別する．

4.2 分離した横断コンサーン

4.1 節で示した非機能特性がコンサーンを規定するものとし，E-AOSAS で定義しているアスペクトを定義するコンサーンを併せると以下に示すコンサーンを識別できる．E-AOSAS に基づくコンサーンとして，

- 並行処理コンサーン
- 状態遷移コンサーン
- アプリケーションロジックコンサーン

が識別でき，RFC の分析に基づくコンサーンとして

- 実時間処理コンサーン
- セキュリティコンサーン
- 例外処理コンサーン

が識別できる．

実時間処理コンサーン

実時間性に関する処理であるタイマ処理は，通信の維持管理で用いられる．送受信処理を行うアプリケーションロジックと関連して，タイマの起動・停止が行われる．実時間処理は，状態遷移コンサーン，アプリケーションロジックコンサーンと横断的に関連する．

セキュリティコンサーン

セキュリティは，上下位層とのデータをやりとりするさいのパケットの暗号化・復号化として実現する．送受信処理を行うアプリケーションロジックと関連して，暗号化処理が行われる．

例外処理コンサーン

パケットの遅延や故障に対応する処理に再送処理がある．再送処理を例外処理とし，通常の送受信処理と関連して行われる．タイマ処理によってパケット遅延の確認をし，時間切れを例外として扱うので実時間処理とも横断的に関連する．パケット復号化の失敗も例外とするのでセキュリティとも横断的に関連する．例外が発生した場合は，例外処理を優先的に行うので，状態遷移やアプリケーションロジックと横断的に関連する．

4.3 アーキテクチャの構築

前節で示したコンサーンをソフトウェアアーキテクチャの構築のために整理すると，以下のアスペクトが得られる．

- 並行処理アスペクト
- 状態遷移アスペクト
- アプリケーションロジックアスペクト
- 実時間処理アスペクト
- セキュリティアスペクト
- 耐故障性アスペクト
- コンフィギュレーションコントロールアスペクト

例外処理をコンフィギュレーションコントロールとして扱う理由は以下のとおりである．

1. 組み込みソフトウェアを並行状態遷移機械の集合として規定
2. イベントが構成を変更する契機である
3. 構成を変更する処理は状態遷移機械として同じ枠組みで説明可能
4. ベースの状態遷移機械の再利用

構成を変更する状態遷移機械は，状態遷移機械を管理する状態遷移機械であるので，メタ状態遷移機械として考える．例外に対応する処理は，アプリケーションロジックの付替えだけで，状態遷移機械の再利用が可能である．

以上により，例外処理をコンフィギュレーションコントロールとして取り扱うことが妥当であると考えた．例外処理として残る処理は，耐故障性に関する処理としてコンサーンを分離し，アスペクトとしてモジュール化する．

構築した TCP/IP 処理ソフトウェアのアスペクト指向アーキテクチャを図 2 に示す．Conduit+ 等の構造に基づき，状態遷移機械で階層構造の整理を行った．TCP/IP 処理ソフトウェアの共通構造として送受信システムがあり，TCP 特有の構造としてコネクション制御システムがある．共通構造にアスペクトを部品として合成した構造となる．

5 アスペクト指向ソフトウェアアーキテクチャスタイルの提案

4.3 節で構築したアーキテクチャをスタイルとして一般化可能かどうかを考える．アスペクトの独立性，アーキ

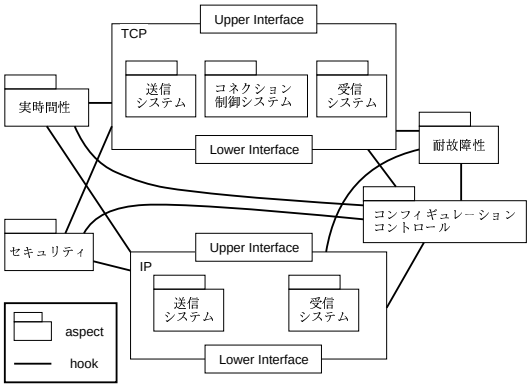

図 2　TCP/IP 処理ソフトウェアのアスペクト指向アーキテクチャ

テクチャの実現に矛盾点がないかを考察する．
組み込みソフトウェアを並行状態遷移機械としてモデル化するので，並行処理，状態遷移，およびアプリケーションロジックアスペクトで構成する．並行状態遷移機械に非機能特性を実現するアスペクトを合成する構造であるので，並行状態遷移機械とアスペクトの関係を整理し，実現について述べる．

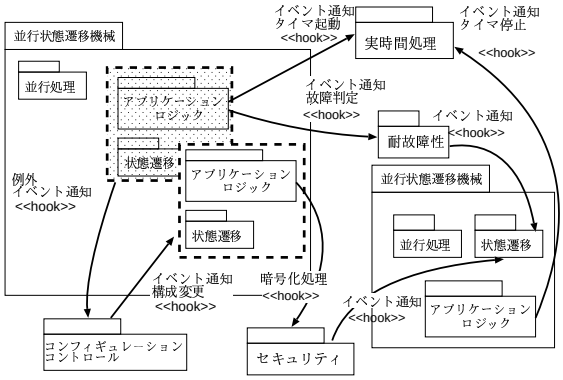

図 3　アスペクト同士の関連

実時間処理アスペクト
アプリケーションロジックの実行前後にタイマを起動・停止させる (図 3，図 4 参照)．実時間処理は，組み込みソフトウェアでは最優先に処理するものと仮定の下，タイマの起動・停止は他のイベントとは特別となる．したがって，並行処理と実時間処理を区別して扱うことができる．時間切れは構成切替えのイベントとして扱い，時間切れ時の処理を別の構成として記述する．

セキュリティアスペクト
データのやり取りを行うシステムではセキュリティを考慮する必要がある．データの暗号化・復号化処理はデータ送受信時に行われる (図 3 参照)．復号化が失敗した場合の処理は，別の構成で記述する．

耐故障性アスペクト
並行状態遷移機械はイベントをやりとりしながら協調動作するので，イベントの通知やデータ送受信時に耐故障を考慮する必要がある．耐故障の判定は，入力機器から出力機器へのイベント通知の耐故障判定を行う (図 3，図 4 参照)．故障のさいの例外処理は別の構成として扱う．

コンフィギュレーションコントロールアスペクト
構成が変更するイベントが発生した場合に，現在起動中のシステムであるベース状態遷移機械から，例外時のベース状態遷移機械に構成を切替える (図 4，図 5 参照)．並行状態遷移機械は状態を常にひとつだけ保持し，コンフィギュレーションコントロールアスペクトが構成変更する．ベース状態遷移機械からメタ状態遷移機械のイベント通知の一実現として try-catch 機構を用いる．構成を切替えるイベントは複数のアスペクトから起こるので，コンフィギュレーションコントロールは，複数のアスペクトと関連する．例えば，時間切れは，実時間処理で発生する例外イベントであるので，実時間処理アスペクトと関連する．

4.3 節で，例外処理をコンフィギュレーションコントロールと耐故障性アスペクトとして扱う理由を述べた．これは，組み込みソフトウェアとしての特性に起因するものであり，通信処理ソフトウェアに起因するものではない．したがって，例外処理をこのように取り扱うことが妥当であると考える．

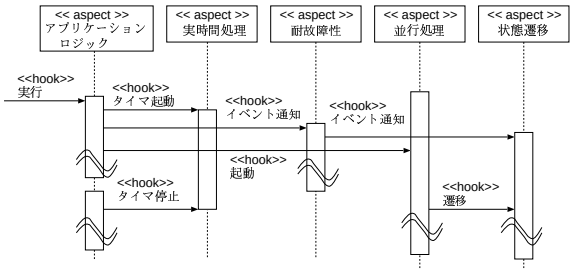

図 4　アーキテクチャの実現

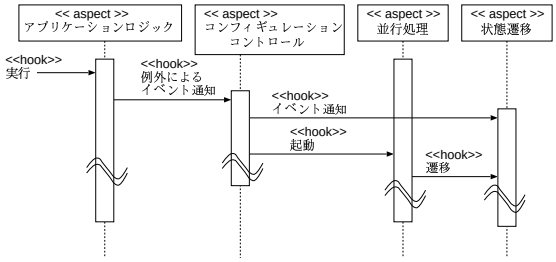

図 5　コンフィギュレーションの切替え

6 考察

本研究で提案した，E-AOSAS+ の応用可能性について考察する．オブジェクト指向による実現との比較で，適正なモジュール化を考察する．また，その他の応用領域で E-AOSAS+ が適用可能かどうかを考察する．

6.1 オブジェクト指向との比較

TCP/IP 処理ソフトウェアでは，RFC で規定される非機能特性に関する処理である再送処理，実時間処理等を実現する必要がある．非機能特性に関する処理は横断的な処理になることを前節までに示してきた．横断的な処理をアスペクトとして分割することで構造の整理が図れることが考察できた．

Conduit+ 等では，コンサーンの分離をデザインパターンを用いて試みていた．状態遷移機械を State パターンと Command パターンによって整理し，遷移と振舞いの部品化を行っている．オブジェクト指向による分離では，遷移のさいの振舞いに関するメッセージ通信が残り，密結合となっていた．アスペクト指向技術を用いることで，部品の疎結合が図られ，部品の再利用が向上すると考えられる．

機能特性と非機能特性をアスペクトとして分離することで，部品化が促進される．このような議論をアーキテクチャ上で行うことで，仕様を整理することができると考える．各部品の組み合わせや，非機能特性の取り扱いの枠組みを与えることは，設計の指針 (仕様) を与えることになると考える．通信プロトコル処理ソフトウェアにおける設計の指針を以下に整理する．

- 基本構造を定義
- 構成を場面ごとに分割して設計
- 非機能特性の取り扱い

コンフィギュレーションコントロールをアスペクトとしてモジュール化したことで，通常の処理と例外時の処理の疎結合を図ることができる．構成を切替える処理をメタ状態遷移機械として定義したことで，ベース状態遷移機械と同じ枠組みで整理可能となった．非機能特性をアスペクトとしてモジュール化し，部品の独立化が可能となったので，設計時に部品の適正な組み合わせが可能となったと考える．

6.2 その他の応用領域への適用

E-AOSAS+ に基づいたアーキテクチャ構築の例として，アプリケーションプロトコル処理ソフトウェアをあげる．アプリケーションプロトコルレベルで，信頼性のある通信を保証する場合は，再送制御や暗号化処理を追加することが考えられる．これらの処理をアスペクトとしてモジュール化したことで，再利用可能な部品として扱うことができる．適切な部品を組み合わせることで，新たな機能の実現が容易であると考えられる．以下の処理が再利用可能であることが確認できる．

- タイマ処理を用いた再送処理
- パケットの破損に対する回復制御 (再送処理)
- パケットの改ざん，盗聴に対する暗号化処理

自動車制御ソフトウェアのような高度に信頼性を必要とする組み込みソフトウェアでは，耐故障性に重点をおいたアーキテクチャの構築が必要である．耐故障アスペクトとしてモジュール化したことで，アーキテクチャ構築時に適正に扱うことが可能となる．

ハードウェア機器が制約される携帯電話制御ソフトウェアでは，動的な構成変更が有効であり，コンフィギュレーションコントロールを重要視した設計となる．E-AOSAS+ に基づいた組み込みソフトウェアのアーキテクチャ構築に有効であると言える．

7 おわりに

本研究では，TCP/IP 処理ソフトウェアを組み込みソフトウェアの典型と位置づけ，アスペクト指向アーキテクチャの構築を行った．構築したアーキテクチャを一般化し，E-AOSAS の改版して E-AOSAS+ の提案を行った．

今後の課題として，E-AOSAS+ に基づいたプロトコル処理ソフトウェアの実現のための開発支援環境を構築することがあげられる．部品の合成時に自動生成可能な部分の抽出と，生成系の開発を行う．

謝辞

本研究を進めるにあたり，熱心な御指導をいただいた野呂昌満教授，有益なアドバイスを下さった張漢明助教授に深く感謝いたします．

参考文献

[1] S. Boking, “ Sockets++ - A Uniform Application Programming Interface for Communication Services, ” *IEEE Comm Mag*, Dec 1996.

[2] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr, and H. Ossher, “ Discussing Aspects of AOP, ” *CACM*, vol. 44, no. 10, pp. 33-38, 2001.

[3] H. Huni, R. Johonson, and R. Engel, “ A Framework for Network Protocol Software, ” in *Proc. OOPSLA ’95*, Oct. 1995, pp. 358-369.

[4] JPNIC,“ JPNIC RFC-JP, ”http://rfc-jp.nic.ad.jp/, Feb 2006.

[5] J. Parissinen and M. Turunen, “ Patterns for Protocol System Architecture, ”in *Proc. Pattern Languages of Programs Conference*, Aug. 2000.

[6] Vine Linux,“ Project Vine, ”http://vinelinux.org/, Feb 2006.