ＵＮＩＸ上でのサービス連携方式の一提案

5Ｔ－１０

－　アプリケーション制御方式　－

伊織 生美　渡辺 一成　川崎 隆二
ＮＴＴ ソフトウェア研究所

１．はじめに

クライアント／サーバモデル（以降Ｃ／Ｓモデルと略記する）においては、多様なアプリケーション（ユーザによる開発〔以下ＵＡＰと略記する〕）の拡張性（追加および変更が容易なこと）を前提に、高性能化およびカスタマジングが重要な課題である。本稿では上記の課題を解決するため、Ｃ／ＳモデルにおけるＵＡＰの会話処理を特徴とするＯＳ（ＵＮＩＸ）上で実装方式に関し特に、①リアルタイムな資源制御 ②アプリケーションの振分け制御 ③アプリケーション間の連携制御 について述べる。

２．Ｃ／Ｓモデルでの位置付け

Ｃ／Ｓモデルの構成を図１に示す。本稿の対

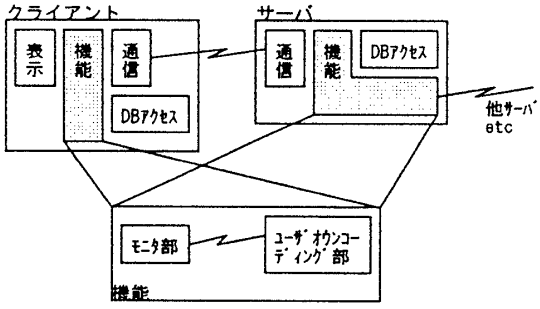

図１ Ｃ／Ｓモデルの構成

象としているＣ／Ｓモデル構成は、図１における機能部分である。ＵＡＰの機能拡張は、サービスの変化に柔軟に対応するため、通常機能部分はモニタ部（モニタ用に共通な制御部分）とＵＡＰに分類される。本稿では、このモニタ部（以降サーバプロセスと略記する）をＵＡＰの処理環境として高性能とカスタマイジングの両方を解決する手段について提案するものである。

A method for real-time and cooperative application services on the client/server environment.
Kiyoshi Iori, Kazunari Watanabe,Ryuji Kawasaki
NTT Software Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180 Japan

３．従来の実装形態の問題

Ｃ／Ｓモデルにおいて、これまでのＵＡＰの独立性、会話処理中心でのユーザ開発環境の自由度増大などの観点から、実装は図２に示す形態ある。サービス要求の発生の都度、モニタ部

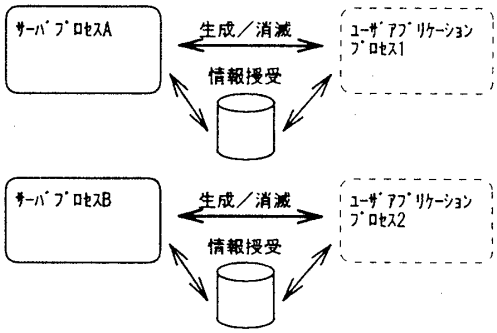

図２ 従来の実装形態

であるサーバプロセスはＵＡＰプロセスを生成し実行し、ＵＡＰの処理終了後、ＵＰＡプロセスは消滅する。このような実装においては、サービス要求とサービスを実現する環境が同期して生成／消滅されるため、ＵＡＰの高速処理、システム内ＵＡＰの資源共有、段階的なシステム拡張のためのカスタマイジング、の各々の実現は困難という問題がある。

３．アプリケーション制御方式

(1) リアルタイムな資源制御

予めＵＡＰの動作環境であるＵＡＰプロセスを起動し、サービス要求が発生した地点でリアルタイムにサーバプロセスとＵＡＰプロセスの間の接続制御を行う。本接続制御によるＵＡＰプロセス時間削減により、高速なＵＡＰの共有化を実現できる。図３にリアルタイムな資源制御方式を示す。図３の接続制御においてサービス要求を受け付け（①）時、サーバプロセスは待ち行列に後に生成する個別通信路用識別詞をＵＡＰプロセスに送信し（②）、ＵＡＰは依頼元サーバプロセスを識別する。その後、サーバプロセスとＵＡＰプロセスとの間に個別通信路を生成（③）し、ＵＡＰの資源占有時間だけ保留する１対１の接続関係が確立される。

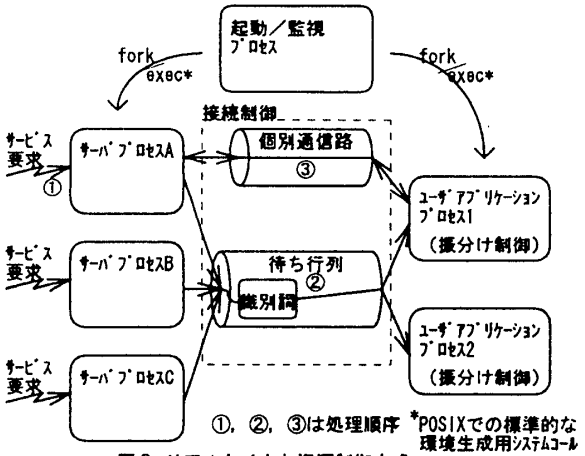

図３ リアルタイムな資源制御方式
（ＵＡＰプロセス群）

(2) 共有型アプリケーション振分け制御

複数のサービス要求から共有されるＵＡＰプロセス内に於は、サービス要求種別（サービス対応のエントリ名）に応じて処理を決定する振分け制御を行う。本制御により個々のＵＡＰ時間の削減とＵＡＰ群の共有が可能となる。図４に振分け制御方式を示す。振分け制御は、個別

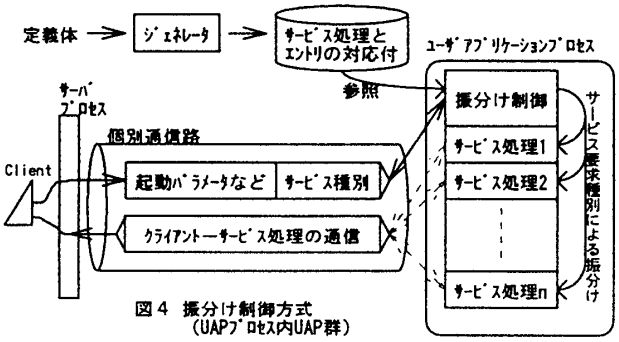

図４ 振分け制御方式
（UAPプロセス内UAP群）

通信路を通してサービス種別、起動パラメータなどを受け取る。振分け制御では起動パラメータを展開し、標準入出力を個別通信路に変更後、サービス種別に対応するサービス処理に振分ける。従って、サービス処理（ＵＡＰ）は、個別通信路を意識することなくクライアントと通信を行う。個々のサービス処理終了後、振分け制御では個別通信路などの資源解放を行い、待ち行列からの次のサービス要求に備える。

(3) アプリケーション間の連携制御

サービス要求毎にＵＡＰを開発する（カスタマイジングに容易なＵＡＰのモジュール化）場合、サービス要求（＝ＵＡＰ）間で情報の引き継ぎをう連携制御が必要となる。連携制御では、ＵＡＰが連携要である指示（ＡＰＩを用意）を行い、サービス処理（ＵＡＰ）終了後、個別通信路の消滅を抑止する。サーバプロセスはＵＡＰプロセスとの１対１の関係を保持している間、接続制御を省略し、生成済の個別通信路を介して複数のサービス要求（＝複数のＵＡＰ）間の連携制御が可能となる。

５．適用効果

サービス要求を依頼してから処理終了までの必要時間についての検討を表１に示す。

表１ サービス要求を依頼から処理終了までの必要時間

	従来の方式での必要時間	本方式での必要時間
起動パラメータを２次記憶装置へ保存	ｔ１	０
ユーザアプリケーションプロセス生成	ｔ２	
ユーザアプリケーションの起動	ｔ３	
起動パラメータを２次記憶装置から読み込み	ｔ４	
処理結果を２次記憶装置に保存	ｔ５	
処理結果を２次記憶装置から読み込み	ｔ６	
起動パラメータ、処理結果を２次記憶装置から消去	ｔ７	
ユーザアプリケーションプロセスの動的結び付け	０	ｔ８
個別通信路生成		ｔ９
個別通信路消滅		ｔ１０

ユーザＡＰプロセスの起動など２次記憶装置に対するアクセスを含む処理は、そうでない処理と比較して多大な時間を要することから、

t1+t2+t3+t4+t5+t6+t7 >> t8+t9+t10

（Σt1〜t2＝数百ｍ秒 Σt8〜t10＝数ｍ秒）

である。

５．おわりに

本稿では、Ｃ／Ｓモデルにおける高性能化、高いカスタマイジングを考慮したアプリケーション制御方式について提案した。今後は具体的なアプリケーションモデルに基づく適用評価と検証を行う。

参考文献

[1]加藤，伊織，川手，長岡：ＵＮＩＸトランザクション処理方式の評価，情報処理学会第４７回全国大会，１９９３−１０
[2]渡辺，伊織，川崎：ＵＮＩＸ上でのサービス連携方式の一提案，情報処理学会第４９回全国大会，５Ｔ−９