(1)モバイル・コンピューティング・プラットフォームAOE

本稿では，USA研究所が提案している，多様な環境条件が時々刻々と変化するモバイルコンピューティングにおいて動作性能およびユーザの使用感を状況変化に適応して維持・向上するコンピューティングプラットフォームであるAOEを紹介する．

Nayeem Islam（ナイーム イスラム）　Dong Zhou（ドン ザオ）
Shahid Shoaib（シャヒード ショアイブ）　片桐 雅二（かたぎり まさじ）

1. まえがき

今日，世界中で大規模な携帯電話ベースの無線データサービスが展開されている．これらの大部分はWebベースのもので，端末側はプログラムで制御できない単純なクライアント（ブラウザ）を用いている．代表的なサービスとして，iモード[1]とWAP（Wireless Application Protocol）[2]の2つが挙げられる．しかし，これらのサービスでは，動作性能（応答時間など）が必ずしも十分ではないこと，ネットワークに接続しなければ操作できないこと，ユーザインタフェース（UI：User Interface）サポートが多くを満足しないこと，ピア・ツー・ピア（P2P：Peer to Peer）アプリケーションを動作できないこと，処理完了の保証ができないことなど，将来のモバイルコンピューティングプラットフォームとしてはさまざまな課題・改善点がある．

USA研究所は，現在の普及しているWebベースのサービス・アプリケーションを活用でき，かつ端末リソースを有効活用してより良い動作性能を実現できるWebアプリケーションを構築するためのプラットフォームの実現を目指している．ここではUSA研究所が提案する次の3点の特徴を持つAOE（Agile Operating Environment）実行環境を紹介する．

(1) 端末におけるサーバ側プログラムコードの適応レプリケーション

移動環境の特徴として，応答待ち時間が長いこと，応答時間がその時によって大きく異なること，切断の可能性が高いことなどが挙げられる．これらに対応するためにAOEでは，従来常にサーバで実行されるアプリケーションプログラムを，部分的にクライアント端末上でキャッシュし実行する．環境の変化に応じてキャッシング判断を実行時に動的に行い，プログラムコードが端末側にキャッシュされている場合，サーバ側にあるプログラムコードあるいはクライアント側にキャッシュされたコピーのどちらを利用するかを動的に選択する．

(2) 端末におけるUIの動的バインド

現在の携帯端末では，そのUI向けの機能・能力は各端末の用途によって均一でなく，Webアプリケーションを作成する上で1つの障害となっている．携帯端末の機能・能力に関係なく，1つのアプリケーションで，すべての機種に対応できることが理想的である．すなわち，能力の異なる端末でアプリケーションを使用する場合には，そのUIが端末に最適な形に自動調整されればよい．これを実現するためにAOEでは，アプリケーション実行時に端末に十分なリソースがあるときには，UIの抽象的記述とその具体的な実装のバインドをクライアント端末で行う．また，このバインドはアプリケーション実行時に動的に行い，その時点で利用可能なリソースに応じて最適な形を選択する．

(3) 適応フォールトトレランス*1に対する端末側サポート

柔軟性に欠けるシステムは，通常1つのフォールトトレランスメカニズムに縛られ，システム状態やアプリケーションからの要求の変化に対応できない．しかし，モバイル環境の特徴は時々刻々と変化する点にあり，1つのフォールトトレランスメカニズムをすべての状況に適応することができない．このため，動的な適応フォールトトレランスが必要となる．障害の無い時と障害回復時の処理を分離し，かつインタフェースによる処理のカプセル化を行うことによりフォールトトレランスメカニズムの動的な変更を行うとともに，端末側に一部処理を分担させる．

本プラットフォームの大きな特徴は，これらの機能により，性能の異なる端末に合わせて動作をカスタマイズしたり，実行時に利用できる資源に応じてユーザやアプリケーションが再構成したり，あるいはシステムが動作を自己適応するようにして，モバイル環境に対応する点にある．

2. AOEアーキテクチャ

AOEでは，クライアント端末からブラウザを介してサービスを要求するWebベースのアプリケーションの利用を前提としている．特に，ユーザに応じて異なったコンテンツを動的に扱うようなアプリケーションを対象としている．

典型的なAOEアプリケーションは，動的にWebページを作成できる一連のMervlet[3]で構成される．Mervletは

＊1　フォールトトレランス：処理中に障害が発生しても，回復処理を行うことによって，矛盾なく処理を継続・完了できるようにすること．対象とする障害の種類により，その対策も異なる．商取引を行うシステムなど重要な処理を扱うサーバ間ではよく用いられている．

Servlet[4]に似ているが，前述の通り，クライアント端末上でWebページを複製・実行できる点，UIを動的に付加する点，クライアント端末やネットワーク，サーバで生じる障害から回復できる点がServletと異なる．

AOE実行環境は，このMervletアプリケーションを実行するための環境である．本プラットフォームでは，まず，クライアントとサーバのそれぞれに，AOE実行環境を展開しておく（図1）．端末のブラウザがHTTP（HyperText Transfer Protocol）リクエストを発行すると，端末上のAOE実行環境（クライアントAOE）により受信され，クライアントAOEは次のいずれかを実行する．

- リクエストをサーバ側のAOE実行環境（サーバAOE）に何らかのトランスポートを使用して送る．ここでサーバおよびクライアントがこのトランスポートをサポートしている場合は，USA研究所の提案する再構成可能メッセージシステム（RMS：Reconfigurable Messaging System）[3]を用いることができる．
- リクエストされたページがローカルで利用可能な場合，あるいはクライアントAOEがリクエストされたMervletをローカルで実行する判断を下した場合，そのリクエストをローカルで実行する．

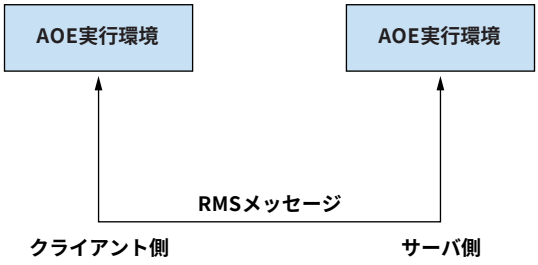

図1　対称型AOEモデル

リクエストに対する応答は，通常，HTML（HyperText Markup Language）などの言語で記述されたアプリケーションからの出力表示である．ただし，状況に応じてクライアント端末向けに作られたUIライブラリにバインドしなおされ，最終的にHTTPフォーマットでブラウザに戻される．

レプリケーションマネージャは，リクエストをどこで実行するかを判断する機能を受け持ち，UIコンポーザがアプリケーションのUIを動的に再バインドする機能を，受け持つ（図2）．また，ブラウザからのHTTP要求については，必要に応じて必ず1度だけは処理・実行するよう，クライアントAOEとサーバAOEが協力して保証される．このような動作は，適応信頼性マネージャが提供する．

基本システム機能に適応性を持たせることが本システムの重要な特徴点である．3つの主要な機能，すなわち，UIコンポーザ，レプリケーションマネージャ，RMSは，環境設定マネージャと機能プロファイラからの入力情報に基づいて適応性判断を下す3つの適応マネージャ（それぞれUIアダプタ，レプリケーションアダプタ，適応信頼性マネージャ）により，適応動作するよう制御される．各機能の適応マネージャは，適応コーディネータによって調整される．各コンポーネントの詳細については，文献[3]を参照されたい．

3. 端末における適応レプリケーション

既存技術にWebキャッシングやプリフェッチ*2技術があ

*2　プリフェッチ：ユーザが今後利用すると思われるデータ（この場合Webコンテンツ）を，リクエストされる前に自発的に読み込んでおく技術．Webコンテンツの構成やユーザのこれまでの使用実績・傾向を手掛かりに読み込むページを決めることが多い．

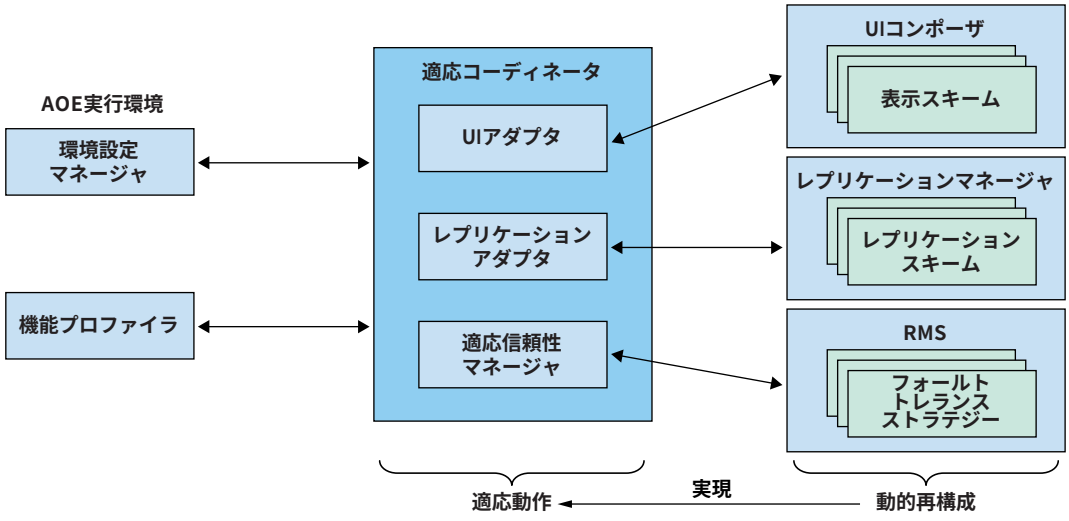

図2　AOE実行環境の構成

る．これらは静的なWebページを作成するモバイル向けWebアプリケーションや一部の動的なアプリケーションに適用できる．しかし，これらの技術は，モバイル向けWebアプリケーションに広く使用されているサーバ側のコードユニット（Servletなど）で生成される動的でパーソナライズされたWebコンテンツに対して，必ずしも有効ではない．AOEでは，サーバ側のコードユニットをクライアント端末に複製することで，このような動的でパーソナライズされたWebコンテンツに対応させる．

Mervletは複製可能なアクティブオブジェクトである．Mervletは，Mervlet自体を定義するクラス，Mervletが使用する読取専用データ，Mervletの書換可能なデータおよびアプリケーション状態で構成される．書換可能なアプリケーションに関するデータは次の2つにグループ分けすることができる．すなわち，1人のユーザまたは1つのセッションが専有するデータと，すべてのユーザまたはセッションが共有するデータである．本システムでは，これらのデータについて，必要最小限の同期によりアプリケーションが要求する一貫性レベルを維持する．

端末，サーバ，およびアプリケーションはそれぞれの立場から，レプリケーションの選択と配置を判断するための条件を定義する．それに応じてAOE実行環境は動的かつ自動的に条件を評価し判断を行う．これにより，レプリカの選択と配置をカスタマイズすることが可能となり，また適応性が実現される．どのレプリカを使用するかの判断は，クライアントがリクエストを受信するごとに，クライアント側とサーバ側のAOE実行環境が連携してその時点の状況に応じて決定する．

4. 端末における動的なUIバインド

ユビキタス環境におけるアプリケーションを考えるとき，さまざまな端末に対してUIをどのように適応させるかは重要な課題である．従来，この問題は出力コンテンツを端末に送る過程で端末能力に適応させるプロキシサーバを使用することで解決できるとされてきた．この代表例として，XML（eXtensible Markup Language）[5]およびXSLT（eXtensible Stylesheet Language Transformations）スタイルシート[6]を使ってコンテンツを書き表す手法がある．しかし，端末が多種多様になると，この手法はXSLTスタイルシートの更新やメンテナンスが頻繁に要求されるため，現実的ではない．さらに，この方式では，端末側の利用可能なリソースをまったく活用することができない．

AOEでは，アプリケーションがXMLを変換・レンダリングし表示する方法をタグ[7]として書き込むようにした．しかし，このタグは静的にアプリケーションにバインドされているわけではなく，動的にバインド可能なライブラリとして構成されるため，実行時に一定の順番でリンクさせることができる．

このUIの動的バインドを実現するために，USA研究所はMSP（Mervlet Server Pages）と呼ぶJSP（Java Server Pages）[4]に似たモデルを提案している．この提案モデルは，JSPライブラリと異なり，開発時にUIをアプリケーションに埋め込まず，UIのプロキシをプログラムに付加し，実行時にUIアダプタが適切なライブラリをバインドして，アプリケーションのUI表示を可能にする．

5. 適応フォールトトレランスの端末側サポート

適応フォールトトレランスのサポートは，RMSとリカバリ可能なMervlet（Recoverable Mervlet）により実現される．

RMSは，Mervlet環境で再構成可能なメッセージ配信機能を提供する．この機能は，RMS-FFI（Reconfigurable Messaging System-Failure Free strategy Interface）と呼ぶインタフェース配下にカプセル化されている．アプリケーションは，このインタフェースに沿ってメソッドを呼び出すだけである．実際に実行されるメソッドは，ユーザまたはアプリケーションの要件に基づいて適応信頼性マネージャ（Adaptive Reliability Manager）が設定する．例えば，RMSはポイント・ツー・ポイント・メッセージサービスを利用するようにも，あるいは集中型メッセージサーバ（JMSなど）を利用するようにも設定することができる．

アプリケーションレベルにおいては，リカバリ可能Mervletがフォールトトレランスを提供する．リカバリ可能Mervletでは，同一のアプリケーションが状況に応じて異なるフォールトトレランスメカニズムを利用することを可能とする．例えば，Webメールアプリケーションでは，ビジネス向け社内電子メールの信頼性を個人向け電子メールよりも高くなるように設定することができる．

主要コンポーネントであるRMSとリカバリ可能Mervletは，無障害時動作設定と障害回復時動作設定が可変である．図3に示すように，これらは適応信頼性マネージャにより動的に設定することができる．無障害時動作設定と障害回復時動作設定を別々に設定できるようにすることで，無障害時動作設定に対応する複数の障害回復時動作設定の開発が容易になる．例えばRMSの場合，メッセージを回復する優先順位をつける障害回復時動作設定とつけない障害回復時動作設定を用意し，切替えることができる．

こうしたフォールトトレランスサポートの適応性を用い

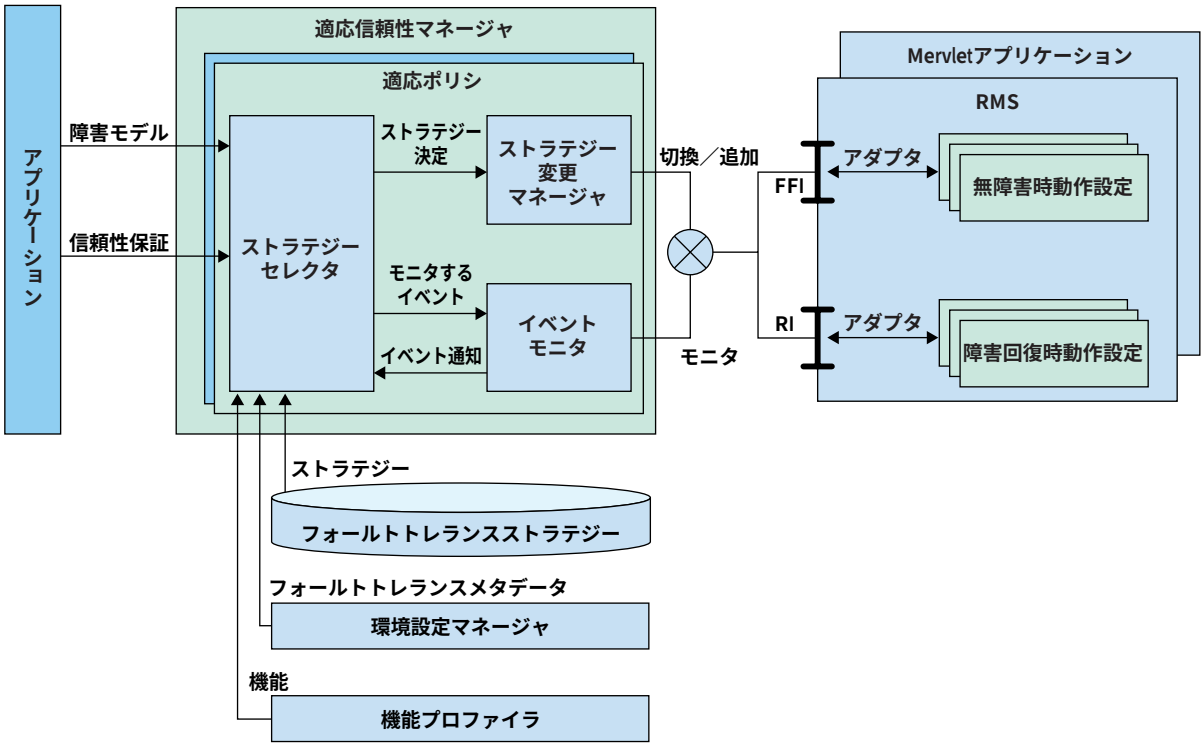

図3　AOEの信頼性サポート

て，サーバ負荷の程度に応じてサーバ側へのロギングを動的にオン/オフできる機能を実装した．サーバ負荷が大きい場合は，適応信頼性マネージャがRMSを再構成し，サーバ側へのロギングを停止させることができる．これにより，特定の状況では応答時間が大幅に改善される．

6. 実装と評価

USA研究所は，AOE実行環境のプロトタイプといくつかのサンプルアプリケーションを実装した．ここでは，このプロトタイプを用いて行った評価実験の結果をいくつか紹介する．

6.1 適応レプリケーションの利点

図4は，WebChessアプリケーションでの次の3つの条件における応答時間の推移を示している．

① サーバ負荷：小，適応レプリケーション：禁止
② サーバ負荷：大，適応レプリケーション：禁止
③ サーバ負荷：大，適応レプリケーション：許可

この実験例では，ステップ番号3の長い応答時間により，ステップ番号4でレプリケーションが実行されている．ここでいうステップとは，事前に定義したチェスアプリケーションのステップである．このグラフから，レプリカの生成・配置とローカルでの起動・実行を合わせた応答時間が，負荷の大きいサーバ上で実行し続けた場合よりも若干

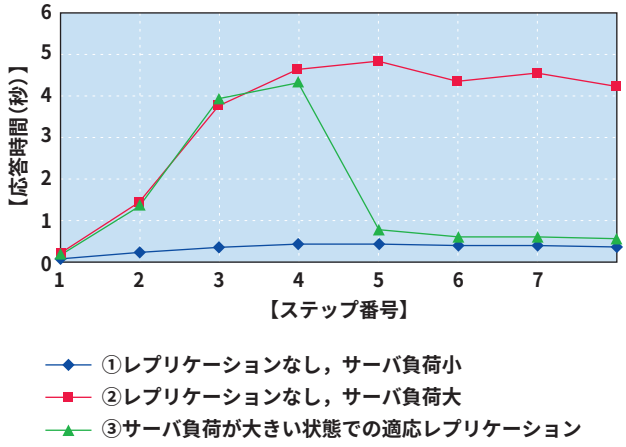

図4　WebChessの応答時間推移（ステップ4でレプリケーション）

（6.6%ほど）短くなったことがわかる（WebChessのレプリケーションに必要な転送ファイルサイズは約37kB）．レプリケーション実行後は，応答時間は大幅に短縮され，サーバ負荷がない状態でのサーバ実行時より若干長い程度であった．

6.2 総合的な適応効果の評価

図5は，Web Calendarアプリケーションにおける，システムの複合的な適応動作による応答時間の改善結果の例を示している．

この実験では，1台当り100人のクライアントを模擬する

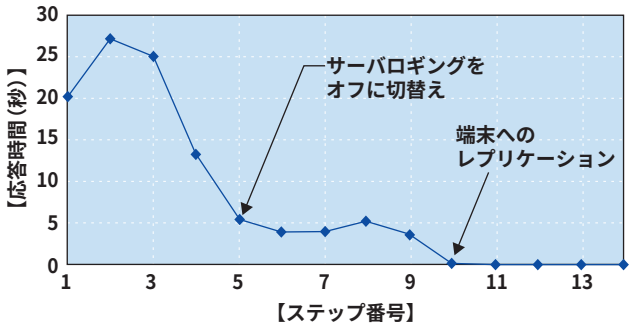

図5　複合的な適応動作の効果

3台のコンピュータからリクエストを発行することによりサーバに負荷を与え，システムの適応動作を観察した．サーバ側のロギングによってサーバ側入出力が増加，サーバ負荷が大きくなり，応答時間が上昇している（20秒以上）．この時点（ステップ番号3）で，システムは適応的にメッセージロギング手法を変更してサーバ側でのメッセージロギングを停止した．その結果，応答時間は5秒程度にまで低下した（ステップ番号5以降）．さらに，システムは応答時間の低減がまだ不十分と判断し，Web Calendarアプリケーション自体を端末側に移した（ステップ番号10）．その結果，応答時間はさらに大幅に短くなった（10ms以下）．この結果から，適応レプリケーションおよび適応フォールトトレランスの動作性能に対する導入効果が確認できる．

7. 関連研究

Rover[8]やOdyssey[9]などは，同様のねらいをもち研究されているシステムである．USA研究所のシステムは，性能と適応性に重点を置いていることがこれらの研究と異なる重要な点である．

サーバ制御によるレプリケーションは，主としてサービスの可用性と信頼性を高めるとともに，サーバ側のロードバランシング（負荷分散）のために利用されている[10]～[12]．これに対して本稿で示した研究では，端末がレプリケーションの各段階を部分的に制御できる点が異なっている．RoverやCodaもオフライン操作をサポートするために端末制御によるレプリケーションを用いているが[8][13]，動的なレプリカ作成および，サービス呼出しやデータアクセスにおける適応性には取り組んでいない．

フォールトトレランスに関しては，信頼できるモバイルアプリケーションを作成するためのツールを提供するRover[14]のような既存の研究と異なり，ロギングをサーバからクライアントに適応的に切り換えることでアプリケーション応答時間を大幅に改善するといった新しい観点を持っている．さらに，複数の障害回復時動作を特定の無障害時動作に対して用意できるユニークなフレームワークを提案している．これは[15]～[18]のような既存の適応フォールトトレランスのシステムには見られないものである．

8. あとがき

モバイル環境は絶えず変化し，それが動作性能およびエンドユーザの使用感に影響をおよぼす．本稿では，状況に適応して動作性能および使用感の低下を軽減するAOEと呼ぶシステムを紹介した．特に，快適なユーザ使用感を実現するために重要な，サービスレプリケーション，再構成可能メッセージ，動的UIバインディングという3つの主要機能について述べた．本システムのポイントは，これら3つの機能を実行時にシステムの状態や端末特性に適応させることができるようにしている点である．また，実行時にシステム内での複数の適応化がいかにして連携しシステムの性能を改善するかについても示した．本稿では，セキュリティの問題には言及していないが，セキュリティも現在取り組んでいる大きな課題の1つである．

文　献

[1] 榎，ほか：“iモードサービス特集，”本誌，Vol. 7，No. 2，pp.6-32，Jul. 1999.
[2] M. Van der Heijden, M. Heijden and M. Taylor: “Understanding WAP: Wireless Applications, Devices and Services,” Artech House, 2002.
[3] N. Islam, D. Zhou, S. Shahid, A. Ismael and S. Kizhakkiniyil: “AOE: A Mobile Operating Environment for Web-based Applications,” To appear in Proc. of IEEE Symposium on Applications and the Internet (SAINT) 2004.
[4] M. Hall: “Core Servlets and JavaServer Pages (JSP),” Prentice Hall PTR, 1st edition, 2000.
[5] E. R. Harold and W.S. Means: “XML in a Nutshell, 2nd Edition,” O’Reilly & Associates
[6] M. Kay: “XSLT : Programmer’s Reference, 2nd Edition,” Wrox, 2001.
[7] B. Shannon, et al: “Java 2 Platform, Enterprise Edition: Platform and Component Specifications,” Addison-Wesley, 2000.
[8] A. D. Joseph, A. F. deLespinasse, J. A. Tauber, D. K. Gifford and M. F. Kaashoek: “Rover: a toolkit for mobile information access,” In Proc. of ACM SOSP-15, 1995.
[9] B. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn and K. R. Walker: “Agile application aware adaptation for mobility,” In Proc. of ACM SOSP-16, 1997.
[10] P. Calton and A. Leff: “Replica Control in Distributed Systems: An Asynchronous Approach,”. In Proc. of 1991 SIGMOD, May 1991.
[11] B. Liskov, A. Adya, M. Castro, M. Day, S. Ghemawat, R. Gruber, U. Maheshwari, A. Myers and L. Shrira: “Safe and Efficient Sharing of Persistent Objects in Thor,”. In Proc. of 1996 SIGMOD, Jun. 1996.
[12] P. Felber and A. Schiper: “Replicating Objects Using the CORBA Event Service?,” In Proc. of FTDCS 97, 1997.

[13] J. J. Kistler and M. Santyanarayanan: “Disconnected Operation in the Coda File System,” ACM Transactions on Computer Systems, 10(1), 3–25, Feb. 1992.

[14] A. Joseph and F. Kaashoek: “Building Reliable Mobile-aware Applications Using the Rover Toolkit,” In Proc. of MOBICOM ’96, Nov. 1996.

[15] I. Chang, M. A. Hiltunen, and R. D. Schlichting: “Affordable Fault Tolerance through Adaptation,” Parallel and Distributed Processing, LNCS 1388, pp. 585–603. Apr. 1998.

[16] C. Sabnis, M. Cukier, J. Ren, P. Rubel, W. H. Sanders, D. E. Bakken, and D. Karr: “Proteus: A Flexible Infrastructure to Implement Adaptive Fault Tolerance in AQuA,” In Proc. of 7th IFIP Working Conference on Dependable Computing for Critical Applications, pp. 137–156, 1999.

[17] N. Venkatasubramanian, M. Deshpande, S. Mohapatra, S. Gutierrez-Nolasco, and J. Wickramasuriya: “Design and Implementation of a Composable Reflective Middleware Framework,” In Proc. of ICDCS 2001.

[18] Z. Kalbarczyk, R. K. Iyer, S. Bagchi, and K. Whisnant: “Chameleon: A Software Infrastructure for Adaptive Fault Tolerance,” IEEE Transactions on Parallel and Distributed Systems, 10(6), pp. 560–579, Jun. 1999.

用 語 一 覧

AOE：Agile Operating Environment
HTML：HyperText Markup Language
HTTP：HyperText Transfer Protocol
JSP：Java Server Pages
MSP：Mervlet Server Pages
P2P：Peer to Peer（ピア・ツー・ピア）
RMS：Reconfigurable Messaging System（再構成可能メッセージシステム）
RMS-FFI：Reconfigurable Messaging System-Failure Free strategy Interface
UI：User Interface（ユーザインタフェース）
WAP：Wireless Application Protocol
XML：eXtensible Markup Language
XSLT：eXtensible Stylesheet Language Transformations