ネットワークによる移動端末支援技術

移動端末は，通信する際の状況により無線リンクが切断されサービスが中断される上，処理能力の制限が大きい．この問題を克服する方法として，移動端末の機能を代理するノードをネットワーク側に配置し，任意の負荷分散処理や切断時処理を可能にする移動端末支援技術である，Twin Agentsアーキテクチャを紹介する．

越智 大介（おち だいすけ）　山崎 憲一（やまざき けんいち）　田中 聡（たなか さとし）

1. まえがき

昨今の移動端末は急速な進化をとげ，性能・容量や機能の向上，無線アクセスの広帯域化，料金の低価格化・定額化が進んでいる．このため，従来では考えられなかった用途やコンテンツが続々と出現しており，今後もその充実と発展が期待できる．このように用途の多様化した移動端末が常時接続される状況下では，移動端末はクライアントとしてサービスを享受するだけでなく，何らかのサービスを提供する側になることも考えられる．つまり，外部の要求から移動端末が情報発信するサーバ型の利用や，移動端末同士が対等に相互接続するP2P（Peer to Peer）型の利用などが増加し，サービスの形態も変化していくと予想される．

しかし移動端末は，通信する際の状況により無線リンクが切断されサービスが中断されやすい．また，PCなどと比較して処理能力の制限が大きいため，提供されるサービスにも制限がある．無線リンクの不安定性を克服し，かつ移動端末の処理能力を補うためには，端末性能や無線技術の進歩および，設備の充実が欠かせないが，それだけでは限界がある．

本稿では，ソフトウェア側からのアプローチでその限界を補う方法として，ネットワークによる移動端末支援技術であるTwin Agents[1]アーキテクチャと，そのプログラミングモデルについて紹介する．Twin Agentsでは，移動端末上の機能を代行する代理ノードをネットワーク側に配置し，代理ノードと移動端末とで協調動作するプログラム実行環境を提供する．代理ノード上のプログラムと移動端末上のプログラムで起動や実行を同期する仕組みや，無線リンク状態またはその他のリソース状態によって代理ノードと移動端末の処理を適応的に変化させる仕組みなどを導入

することにより，移動端末のユーザは任意のプログラムの負荷分散処理や切断時処理を実現することができるようになる．

以下より，Twin Agents アーキテクチャの詳細およびその応用例について説明する．

2. Twin Agents アーキテクチャの要求条件

移動端末は，その独自の特性のため，将来の多様なサービス形態を実現するにあたって，以下のような制限がある．

・移動通信の特性上，状況により無線リンクが不安定になり最悪の場合，切断される場合がある．また，圏外エリアを完全に無くすことは難しい．

・PC と比較して処理能力が低いため，利用できるコンテンツやサービスに制限がある．また，サーバ型やP2P型として動作することが難しい．

上記の問題を克服するために，Twin Agents が満足すべき主要な要求条件を以下に示す．

(1) 通信コネクションの存続性

通信の安定性のため，移動端末および通信中の相手ノードで動作するアプリケーションは，無線リンクの切断などに依存せずに通信可能である．実際には無線リンク切断中は通信不能であるが，通信コネクションを維持することで，アプリケーションに対して透過的に通信を維持する．

(2) 切断時処理

無線リンクが切断した場合においても，切断の影響を低減して通信を継続するために，移動端末および通信中の通信相手ノードは，切断時の処理により何らかの代替サービスを享受することができる．

(3) サービスとリソースの非依存性

移動端末は，その処理能力に依存せずにサービスを提供・享受できる．

以上の要求条件を満足するために，Twin Agents アーキテクチャでは，移動端末以外のノードにて，通信コネクションの存続性および切断時処理によるサービス継続性を確保し，移動端末の処理能力を代替する方法を考える．

3. Twin Agents アーキテクチャの設計

2章の要求条件を満足するために，Twin Agents は移動端末に専属の代理ノードをネットワーク上に配置し，その代理ノードと移動端末との間で分散処理を行う仕組みを提供する（**図1**）．代理ノードは，安定性の高いコアネットワークと比較的不安定な無線ネットワークとの中間に配置され，移動端末との連携に必要なプログラム実行環境とデータのストレージ機能（蓄積機能）を提供する．移動端末は，代理ノードを経由して通信相手ノードと通信をする．代理ノードを導入することで，無線リンクの切断時などに，移動端末に代わって代理ノードが通信を続行することができ，また，移動端末の処理の一部を代理ノードが代行することで，移動端末の処理能力を補うよう連携することができる．

代理ノードと移動端末の連携は，双方にインストールさ

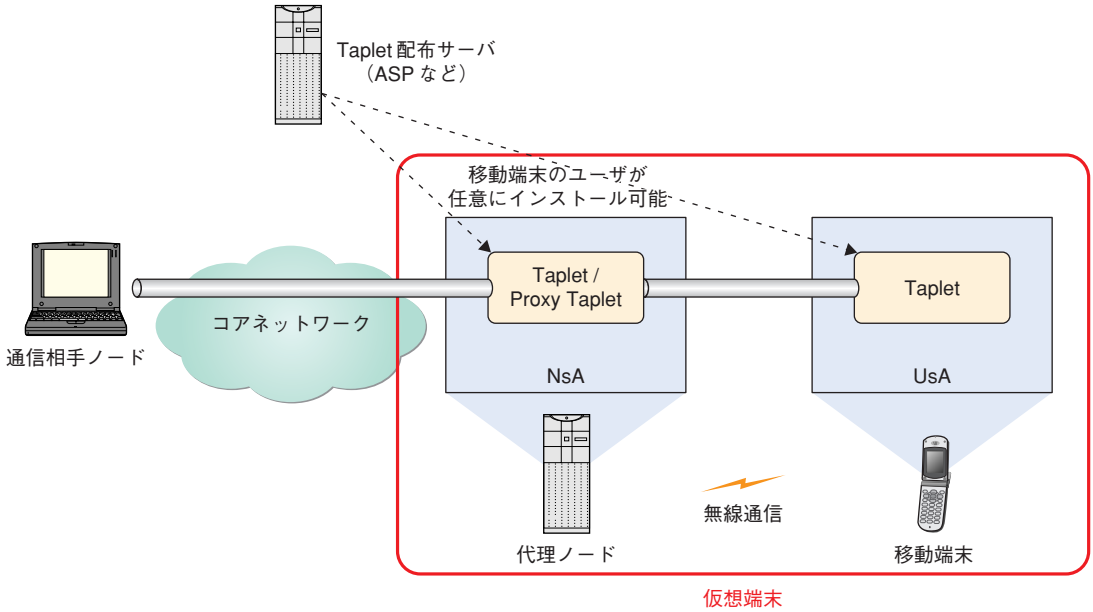

図1　Twin Agents の概要

れるミドルウェアで実現される．代理ノード側のミドルウェアをNsA（Network side Agent），移動端末側をUsA（User side Agent）と呼ぶ．このNsAとUsAは1つの仮想端末を構成し，通信相手ノードは基本的にNsAとUsAを区別せず，仮想端末を通信先として認識する．

NsAとUsAはそれぞれ任意のプログラムを実行可能である．その実行環境で動作するプログラムをTapletと呼ぶ．Tapletは，個人やASP（Application Service Provider）などがプログラミング可能である．Tapletには，NsA内のTapletとUsA内のTapletの2つがあり，個人やASPなどのプログラマはアプリケーションごとにこの2つのTapletのペアを作成する．これらTapletは，移動端末のユーザが任意にインストール・実行できる．

それぞれのTapletは，NsAとUsAの間のリンクが確立しているときは協調して動作し（協調モード），切断された場合は自律的に動作する（自律モード）．リンク切断時にTapletの動作を協調モードから自律モードへ切り替えることで，通信相手ノードおよび移動端末のユーザに対して代替的に処理を継続させることができる．これにより，不安定な無線リンクの切断への対処をNsAとUsAの間で閉じることができ，通信相手ノードは移動端末側の無線の不安定性を意識することなく通信することが可能となる．これは2章の要求条件(1)，(2)に対応する．

また，Tapletには，代理ノード上のストレージやCPU（Central Processing Unit）を利用して移動端末の処理能力を補うための分散処理が記述可能である．これにより，通信相手ノードは移動端末の処理能力を考慮することなく通信が可能となる．これは，2章の要求条件(3)に対応する．NsAにサーバ型プログラムを配置することで，リンク切断時にも仮想端末がサーバ型の動作をすることも可能となる．

さらには，リンク状態以外にも，移動端末のその他のリソース状態（例えばバッテリ残量など）に応じてTapletの動作を切り替えることも可能である．

以上のような特長をTwin Agentsは備える．そしてそれらの特長をNsAとUsAにて実現するため，通信相手ノードを改変する必要がない．

4. 技術課題と各機能

3章のようなTwin Agentsを実現する上で，次の技術的課題がある．

①代理ノードと移動端末に分離したプログラムを同期して実行する方法．

②実コネクションが切断していてもこれを隠蔽すると同時に，必要に応じてアプリケーションに開示することを，従来のソケットインタフェースを大きく変更することなく提供する方法．

③代理ノードのNsAでも協調モード・自律モードに応じて動作を変更させるため，プログラムが複雑になりがちであるが，そのプログラム記述を容易にする方法．

④データ送信途中に通信が切断した場合にも，NsAとUsAが矛盾しない状態を保つ方法．

以下では，Twin Agentsの各構成要素について，上記の技術課題を中心に説明する．また，**図2**にアーキテクチャの概要を示す．

(1) アプリケーションマネージャ

アプリケーションマネージャは，NsAとUsAにTapletをインストールし，Tapletのバージョンを管理する．また，Tapletの実行を管理し，NsA内とUsA内の2つのTaplet間で同時起動・同時終了を管理する．技術課題の①を解決するために，NsA，UsAの持つTaplet管理情報を相互に交換し，起動状態やバージョンの不一致を防ぐ．

(2) コネクションマネージャ

コネクションマネージャは，NsAとUsAの間で確立されるコネクションや，通信相手ノードとNsAの間で確立されるコネクションの生成・消滅などを管理する．また，後に示すPSID（Persistent Socket ID）の管理も行う．

(3) リンク状態によって動作を切り替えるソケット（Persistent Socket）

Persistent Socketは，協調モードと自律モードを切り替えて切断時処理に対応するための通信ソケットである．**図3**にこの概要を示す．これはNsAとUsAの間でのみ生成される通信ソケットで，リンク切断時にもTapletに対して仮想的なコネクションを維持し，NsAとUsAの間の実コネクションの再接続などの処理を自動化する．

Twin Agentsでは，NsAとUsAの間のコネクションの識別子に，下位の通信プロトコル（例えばTCP/IP（Transmission Control Protocol/Internet Protocol）など）に依存しない独自のPSIDを使う．これにより，NsAとUsAの間のリンクの切断，移動端末のネットワークインタフェースの変更・消滅，移動端末のアドレスの変更といった，種々のリンク状態の変化から，Tapletのコネクションを分離することができる．

また，Persistent Socketが規定するプログラミングモデルを用いて協調モードと自律モードを記述し，リンク状態に応じてTapletの動作を切り替えることが可能である．さらに各モードには，モードが持続している時（持続時）の処理と，モードが切り替わった時（変化時）の処理を

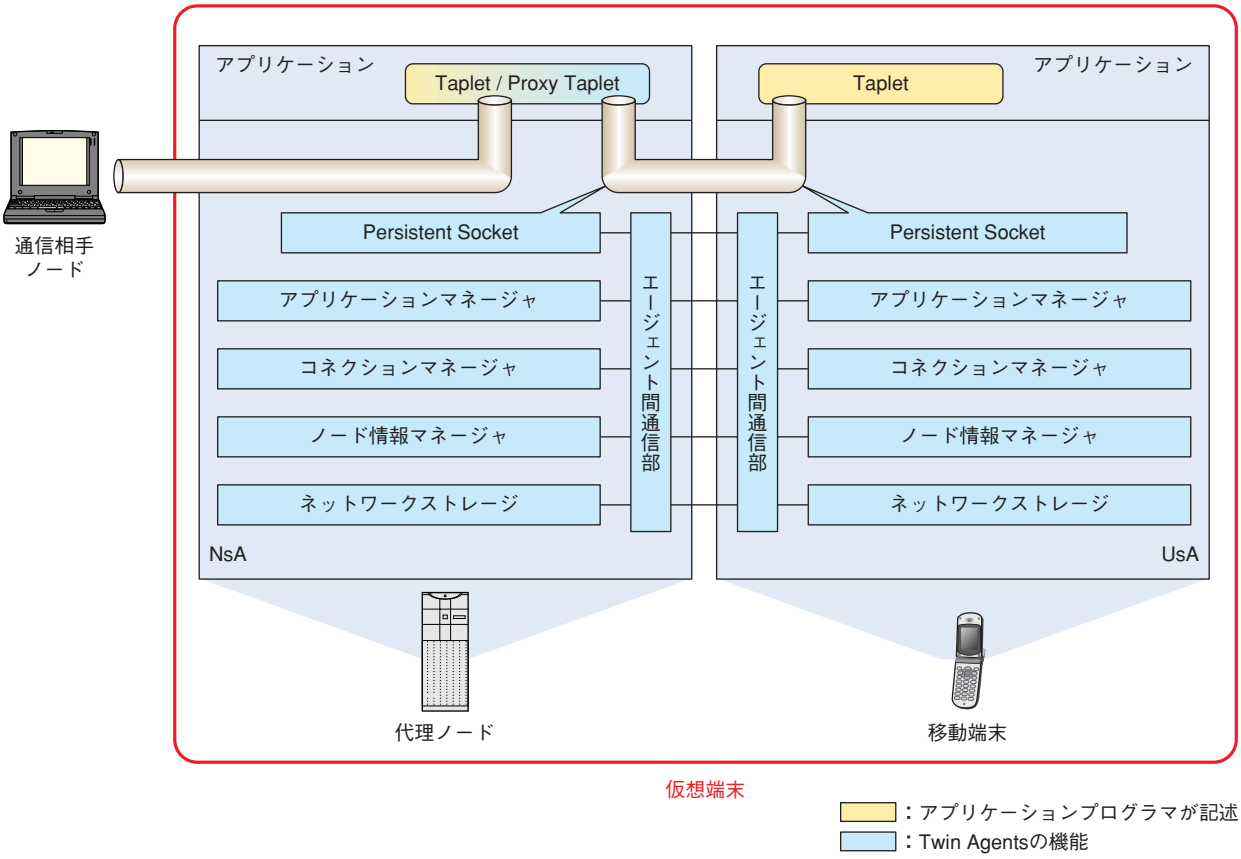

図2　Twin Agentsのアーキテクチャ概要

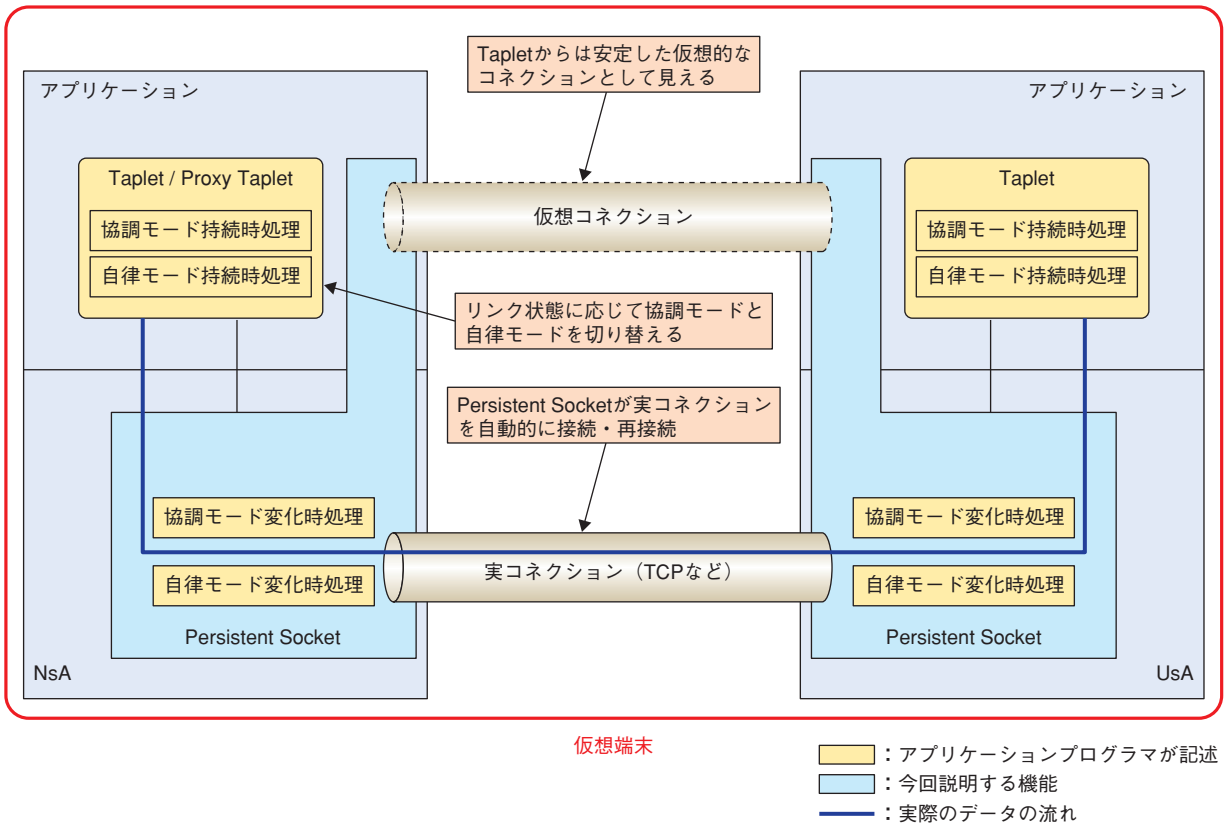

図3　Persistent Socketの概要

個別に記述できる．これにより，アプリケーションに共通して自動化される部分とアプリケーション固有の部分が明確になり，Taplet での切断時処理の記述が容易になる．さらに Persistent Socket は Socket と呼ばれる汎用的な通信プログラムモジュールを拡張しているため，通常の Socket と同様に扱うことも可能である．

以上により，Persistent Socket は技術課題の②を解決する．

(4) 拡張可能な基本プロキシの雛形（Proxy Taplet）

一般的にプロキシとは通信経路の途中のノードでデータを蓄積・変換し，転送する機能を指す．NsA 内の Taplet は，UsA 内の Taplet に対してプロキシ的な役割を担う場合が多々ある．Proxy Taplet は Taplet を拡張して共通的なプロキシ機能を備えたプログラムモジュールで，NsA 内のプログラムがデータを変換し転送するために必要な共通機能と，アプリケーションに依存する部分の記述モデルを提供する．

共通機能には，通信相手ノードと UsA のデータを相互に受け渡す機能，入力されるデータの内容を判断してデータを加工する機能などがある．Proxy Taplet を利用し拡張することで，これらの機能を容易に利用できる．これにより，技術課題の③を解決する．

図 4 は Proxy Taplet の概要である．UsA 内の Taplet は，Proxy Connector というプログラムモジュールを経由して NsA 内に Proxy Taplet，もしくはそれを拡張したプログラムモジュールを起動し，通信相手ノードまでの通信コネクションを確保できる．プログラマは，Proxy Taplet 内の転送時加工処理と，Proxy Connector 内の送受信時加工処理を任意に拡張可能である．これにより，Proxy Taplet である種の符号化を行い，それに対応する復号化を Proxy Connector で行うことが可能となる．

また Twin Agents では，機能の実装レベルに応じた複数の Proxy Taplet を用意している．例えば Unit Proxy Taplet は，Proxy Taplet を拡張して一定単位の通信を保証する機能を付け加えた汎用的な Proxy Taplet である．これらを用いることで，技術課題の④を解決する．プログラマは目的に応じて Proxy Taplet を選択し，任意に拡張できる．

(5) ノード情報マネージャ

ノード情報マネージャは，代理ノードおよび移動端末の静的／動的リソース情報を管理する．静的リソース情報としては，CPU 能力，ストレージ容量，ネットワークメディア情報，入出力リソースの情報（表示解像度，入出力装置など），電源の種類などがある．動的リソース情報としては，CPU 負荷，ストレージ残量，通信の状態（ネットワーク負荷，電波強度，通信モードなど），電池残量，移動端末の位置情報，移動端末の状態（マナーモードなど）などがある．これにより，Taplet ではさまざまな状況に応じて動作を変更可能で，例えば移動端末のバッテリ残量に応じて Taplet の動作を変更することも可能である．

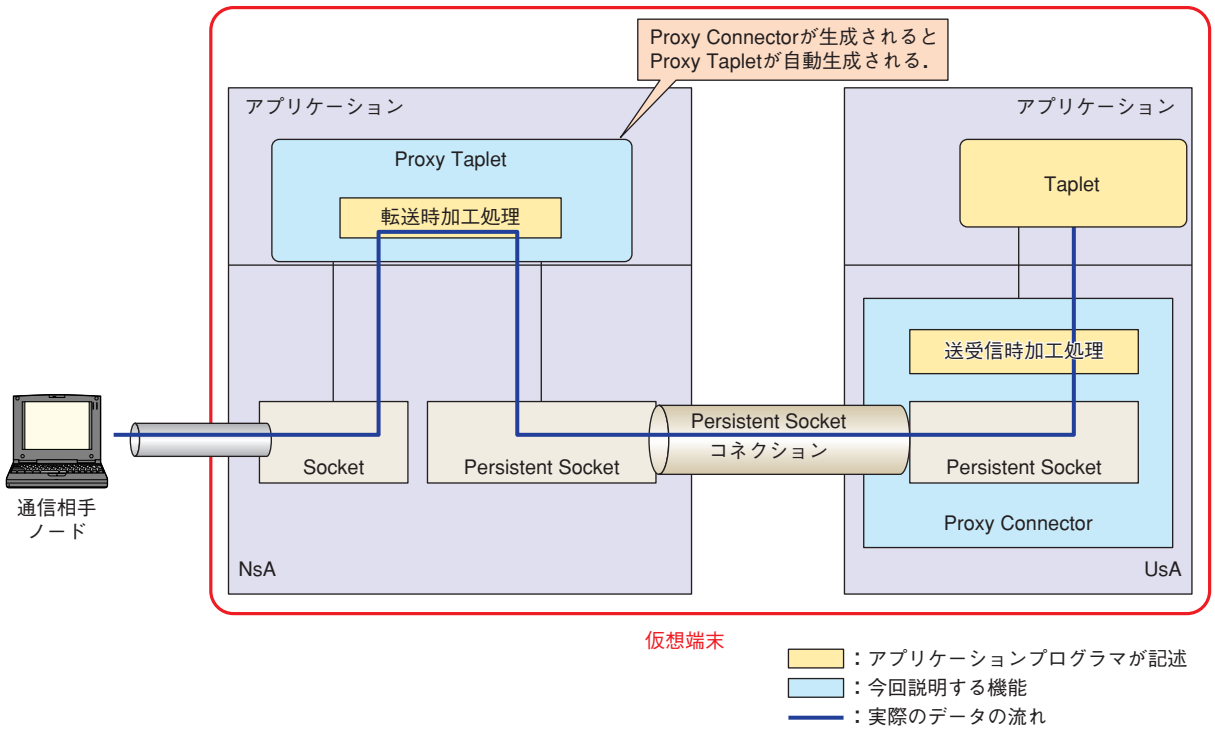

図 4　Proxy Taplet の概要

(6) ネットワークストレージ

NsAとUsAは1つの仮想端末を形成するため，基本的に共通の情報を保持している必要がある．そこで，NsA内のプログラムもUsA内のプログラムも共通のアクセス手法でアクセスし，データの一貫性を確保するストレージとして，ネットワークストレージを提供する．

Twin Agentsは以上の機能群により構成される．これにより，無線リンクの不安定性を克服し，移動端末の処理能力の低さを補う仕組みをTapletに提供する．

ここでは，代理ノードに1つのNsAを配置することを前提に説明したが，複数のNsAを配置することも可能である．

5. Twin Agentsの応用例とその評価

Twin Agentsは，特定のプロトコルに依存しないため，切断時処理や分散処理を考慮したさまざまなアプリケーションを実現できる幅広い汎用性を持つ．そして，あらかじめ用意されたツール群が低レベルAPI（Application Programming Interface）から高レベルAPIまで幅広く提供するため，複雑な処理を比較的容易に作成することが可能である．ここでは，Twin Agentsで実現可能な代表的なアプリケーションを分類し，その一例であるビデオ会議アプリケーションをもとに評価する．

5.1 アプリケーションの分類

Twin Agentsは移動端末の能力をアシストするプラットフォームであり，単に切断時処理を提供する以外にも複数の利用用途がある．Twin Agentsを利用したアプリケーションは，以下のようないくつかのアプリケーションタイプに分類できる．また，Twin Agentsが対象とすべきアプリケーションは，これらのアプリケーションタイプの複合型である場合が多い．

(1) 切断時処理タイプ

移動端末のリンクが切断中に，または切断に備えて，中継ノードなどの代理ノードで何らかの処理をするタイプ．このタイプはさらに細分される．例えば，移動端末に代わって通信相手ノードに代理応答するタイプ（例：圏外トーキー，Web），通信を別のノードへ転送するタイプ（例：電話転送サービス，メール転送サービス），通信を蓄積保存するタイプ（例：伝言サービス，着信履歴保存サービス），などがある．

(2) 情報蓄積タイプ

通信相手ノードと移動端末の間の通信内容や移動端末などのノード情報を，代理ノードにて蓄積するタイプ．蓄積した履歴を後から利用するようなケースが考えられる．例えば蓄積した通話内容を後から再生するといった応用例がある．

(3) 通信加工タイプ

通信相手ノードと移動端末の間の通信を代理ノードにて加工し，通信に付加的要素を加えるタイプ．例としては暗号化，通信経路多重化，メディア変換などがある．

(4) 代理処理タイプ

移動端末が処理すべきタスクを，移動端末の負荷分散やコスト低減を目的として，代理ノードが処理するタイプ．例としては代理認証，代理情報収集，代理情報発信などが考えられる．移動端末がUI（User Interface）部分のみを担当する場合，それは移動端末が代理ノードのコントローラであると見なせる．

上記のアプリケーションタイプはすべてTwin Agentsを用いることで実現可能である．これらの要素を複合的に持ち得るアプリケーションとしては，ビデオ会議アプリケーションや，オンラインリアルタイムゲームなどがあり，これらはTwin Agentsを利用することで実現が容易になると考えられる．

5.2 ビデオ会議アプリケーションの実装

Twin Agentsのメリットを示す一例として，3者間におけるビデオ会議アプリケーションを作成した．ネットワーク構成を**図5**に示す．3者はP2Pで接続され，そのうち1者がTwin Agents上（仮想端末上）に実装されている．通信相手ノードA，Bは切断時処理を考慮していない従来のビデオ会議アプリケーションである．

Twin Agentsで動作するビデオ会議アプリケーションはNsA内とUsA内にそれぞれTapletを用意する．Twin Agentsでは，受け取ったデータの蓄積が可能なProxy TapletであるBuffering Proxy Tapletを用意しているが，NsA内のTapletはそれを拡張し，切断時の代理応答機能を実装している．

Twin Agentsを利用したビデオ会議アプリケーションでは，従来のビデオ会議アプリケーションでは実現困難な次の動作が実現可能となった．

(1) リンク切断時（自律モード持続時）の動作

- NsAが，通信相手ノードA，Bとのコネクションを維持する．
- NsAが，通信相手ノードA，Bから送信されるマルチメディア情報（映像・音声・文字チャット・その他の状態）のすべてを蓄積する．

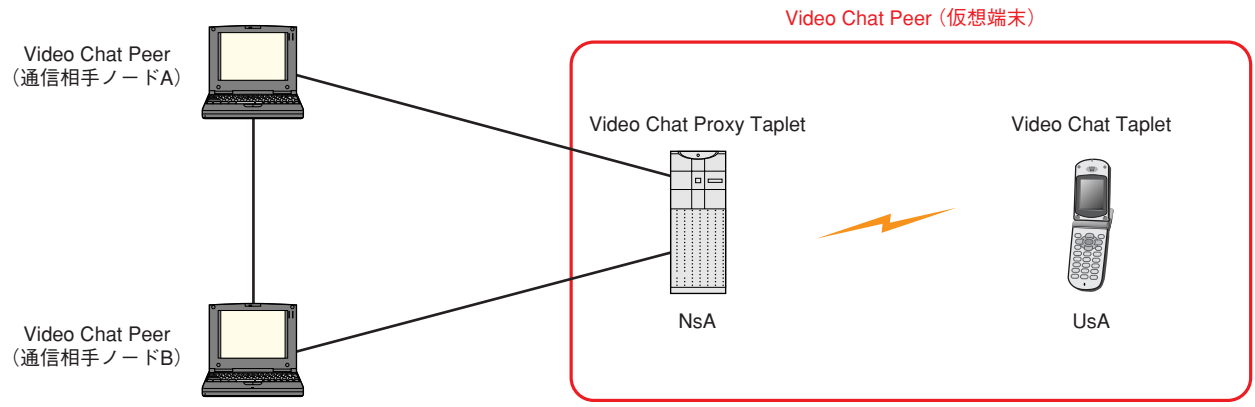

図5　ビデオ会議アプリケーションのネットワーク構成

・NsAが，通信相手ノードA，Bに対してユーザの画像に代わる代理映像を送信する．
・NsAが，通信相手ノードA，Bからの（スケジュールなどの）問い合わせに，移動端末のユーザに代わって代理応答する．
・UsAが，移動端末のユーザからのマルチメディア情報の入力を蓄積する．

(2) リンク復活時（協調モード変化時）の動作

・UsAとNsAが蓄積したマルチメディア情報を相互に通知し，早送り再生などを行い同期する．

これにより仮想端末のユーザおよび通信相手ノードA，Bのユーザは，会議の内容を失うことなく，会議を続行することが可能となる．

またTwin Agentsは，切断時のアシストだけでなく，通常時も移動端末をアシストする．例えば，通信相手ノードA，Bから送られてくる映像・音声を合成し，データサイズを削減して移動端末へ送信するほか，移動端末が対応するデータフォーマットへ変換する．また通信データをリアルタイムで大量に録画することも可能である．メディア変換をNsAで実現するために，通信相手ノードは移動端末の機能制限を意識することなく汎用的なフォーマットで通信できる．

以上により，Twin Agentsを用いることで，少なくとも前述の切断時処理タイプ，情報蓄積タイプ，通信加工タイプが実現可能であることが分かった．

また，ビデオ会議アプリケーションの実装において，Persistent SocketおよびProxy Taplet の利用により，それらを利用せずに独自に処理を記述した場合と比較して，通信部分およびプロキシ部分のコード量が約20％削減されることを確認した．また，別の評価用アプリケーションとして作成したIRC（Internet Relay Chat protocol)[2]クライアントソフトにおいても，約30％のコード量が削減されることを確認した．これにより，Persistent SocketおよびProxy Taplet が複数のアプリケーションへ適用可能で，プログラミングを容易にすることが確認された．

6. 関連研究との比較

Rover[3]，Odyssey[4]およびMervlet[5]は，移動透過型および移動認識型アプリケーションを構築するためのツールであり，Twin Agentsと目的が似ている．しかしこれらは，end-to-end型のシステムであるため，サーバ側の対応が必要である．さらにサーバ制御の処理コードをクライアントに移動して処理・同期するシステムであるため，基本的にサーバ制御の切断時処理のみを対象とし，クライアント制御の切断時処理を実現できない．また，移動端末はクライアントとして動作することを前提とする．これに対しTwin Agentsは，クライアント制御の切断時処理が可能で，移動端末はサーバとして動作することが可能である．

また，WWW（World Wide Web）情報発信システム[6]は，移動端末がネットワークから切断されても安定して情報を提供する仕組みを提案している．ただ，提供する情報がWebに限定されている点で柔軟な切断時処理を実行できない．Twin Agentsは，Webに限らず，仮想端末にて安定して多種多様なサービスを柔軟に提供できる．

7. あとがき

本稿では，移動端末上でサービスを提供・享受する上で問題となる無線リンクの不安定性と移動端末の低い処理能力に関する問題を解決するため，状況に応じた柔軟な切断時処理と，代理ノードと移動端末間での分散処理が実現可能なTwin Agentsアーキテクチャについて述べた．本アーキテクチャでは，リンク切断時の影響を隠蔽し，切断時処理を記述可能なPersistent Socketと，基本プロキシ機能の

動作を隠蔽したProxy Tapletの利用により，代理ノードと移動端末との分散処理プログラムの記述を容易にしている．本アーキテクチャは，移動端末に対してだけでなく，ダイヤルアップ接続のデスクトップPCのような，移動しないノードに対して適用した場合にも有効である．

Twin Agentsは切断時のアシストだけでなく，代理処理・通信加工・情報蓄積などのサービス形態においても利用可能であり，移動端末の能力をネットワーク上の代理ノードが強力にアシストすることにより移動端末の持つリソースの制限を緩和させ，移動端末のユーザの利便性を向上させることができる．

今後は，来たるユビキタスコンピューティング[7]環境を想定し，代理ノードと移動端末の間だけでなく，その周囲にある情報家電やセンサネットワークなどを取り込み，移動端末より非力なセンサなどをアシストする適応的アプリケーション環境に拡張する予定である．

文　献

[1] D.Ochi and K.Yamazaki:“Twin Agents: Network-assisted Disconnected Operation and Distributed Processing for Mobile Communication,”Proc.Of IEEE Symposium on Applications and the Internet 2004 (SAINT2004), Jan.2004.

[2] J.Oikarinen and D.Reed:“Internet Relay Chat Protocol,”RFC 1459, May 1993.

[3] A.D.Joseph, J.A.Tauber and M.F.Kaashoek:“Mobile computing with the Rover toolkit,”IEEE Transactions on Computers, Special Issue on Mobile Computing, Vol.46, No.3, pp.337-352, Mar.1997.

[4] B.D.Noble, M.Satyanarayanan, D.Narayanan, J.E.Tilton, J.Flinn and K.R.Walker:“Agile Application-Aware Adaptation for Mobility,”Proc. the 16th ACM Symposium on Operating System Principles (SOSP-16), pp.276-287, Oct.1997.

[5] N.Islam, D.Zhou, S.Shahid, A.Ismael and S.Kizhakkiniyil:“AOE: A Mobile Operating Environment for Web-based Applications,”Proc.Of IEEE Symposium on Applications and the Internet 2004 (SAINT2004), Jan.2004.

[6] S.Tagashira, et al.:“Adapting a Mobile Information Announcement System for Network Connectivity,”Proc. 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA’99), Vol.2, pp.970-976, Jun.1999.

[7] Mark Weiser:“The computer for the 21st century,”Scientific American, pp.94-104, Sep.1991.

用 語 一 覧

API：Application Programming Interface
ASP：Application Service Provider
CPU：Central Processing Unit
IRC：Internet Relay Chat protocol
NsA：Network side Agent
P2P：Peer to Peer
PSID：Persistent Socket ID
TCP/IP：Transmission Control Protocol/Internet Protocol
UI：User Interface
UsA：User side Agent
WWW：World Wide Web