動的な環境に適応するシームレスなサービス連続技術

高杉耕一[†1] 中村元紀[†1] 田中聡[†2]
久保田稔[†3] 小柳恵一[†4]

ネットワークトポロジの動的な変化やユーザのネットワーク上の移動が頻繁に起こるようなユビキタスネットワーク環境において，シームレスにサービスを連続させる技術を提案する．将来，多種多様な端末があらゆる所でユビキタスネットワーク環境を構成し，端末やユーザは自由に移動することが考えられる．つまり，ネットワークは動的に変化し，分割，融合し，ユーザもこのようなネットワーク上を自由に移動する．そこで，このような環境の変化をともなった場合においても，同一サービスを中断することなく連続的に利用したいという要求が発生する．我々はこの要求を満たすため，シームレスにサービスを連続させるプラットフォームを構築した．このプラットフォームはトランスポート層の上位に通信を中継するプロキシを配置し，ネットワークの変化を隠蔽したり，利用端末の変更等の要求を実現したりする．さらに，このプラットフォームは OS 上位のミドルウェアで実現しているため，PC，PDA 等ユビキタスネットワーク環境で想定される多種多様な種類の端末で動作させることが可能である．本論文では上記の提案を行うとともに，実装したプロトタイプの評価実験により，環境の変化に追従する切替え時間，プラットフォーム利用による転送速度，遅延時間を測定した．その結果，LAN 環境で切替えにともなう通信の中断時間は 1.5 秒以内，データの転送速度の増加は 8% 以内，遅延時間の増加は 3.5 倍程度と，アプリケーションレベルのプラットフォームでも十分実用的に動作可能であることを確認した．

Technology for Service Continuity Adapting to Dynamic Environment Seamlessly

Koichi Takasugi,[†1] Motonori Nakamura,[†1] Satoshi Tanaka,[†2]
Minoru Kubota[†3] and Keiichi Koyanagi[†4]

This paper proposes a seamless service platform that continuously provides services in a dynamic network environment. In the future, many kinds of terminals will be used to form a ubiquitous network environment anywhere. This type of network can dynamically change, merge, or separate, and users migrate on their networks. Hence, concurrent with the environment change many requests are made. Our proposed overlay network is a seamless services platform that conceals many kinds of changes and responds to the user's requests. This platform can easily accommodate existing applications and application protocols, because a common function can be reused by existing applications and application protocols. Moreover, this platform is based on middleware over the transport layer (such as TCP/IP). Since it is OS-independent, it is easy to support many kinds of terminal devices, such as PCs and PDAs. Experiments with prototype implementation of our seamless service platform demonstrated its effectiveness and feasibility.

1. まえがき

近年，PC や PDA をはじめとするさまざまな端末（ノード）の小型化や多様化，無線技術や通信媒体の多様化，高度化が進展してきた．それにともない，多種多様な端末がいつでもどこでもネットワークに接続されるネットワーク（ユビキタスネットワーク）への期待が高まっている．

ユビキタスネットワーク環境では，ノードがネットワーク内を頻繁に移動することで，ネットワークのト

†1 日本電信電話株式会社 NTT 未来ねっと研究所
NTT Network Innovation Laboratories, NTT Corporation
†2 NTTドコモ株式会社ネットワーク研究所
Network Laboratories, NTT DoCoMo Inc.
†3 千葉工業大学電気電子情報工学科
Chiba Institute of Technology
†4 早稲田大学情報生産システム研究科
Graduate School of Information, Production and Systems, Waseda University

ポロジは動的に変化し，ノードのネットワークとの接続点も変化する．さらに，この環境上を自由に移動するユーザがいつでも，どこでも，シームレスに同一サービスを享受しつづけたいという要求が高まっている．たとえば，ユーザが屋内から屋外に移動し，ユーザは屋内の PC から処理能力は低いが携帯性に優れた PDA に切り替えて同じサービスを利用するといった状況が考えられる．このように，変化する環境や要求に適応可能なネットワーキングサービスが求められている．本論文において，シームレスにサービスを連続させるということはネットワークトポロジやノードの変化に対し，サービスや通信の状態をつねに引き継ぎつつ，サービスを継続させることを示す．このとき，ユーザの操作性，ユーザに対する表示等を継目（ユーザは再度同じ操作を要求されたり，必要以上に過去の表示を重複して見たりすること）なく引き継ぐ．また，このネットワーキングサービスの適用範囲を広げるという観点から，専用の新しいアプリケーションによるサービスを想定するのではなく，既存のアプリケーションやアプリケーションプロトコルをそのまま使えるようにすることが望ましい．また，これらのネットワーキングサービスを多種多様な OS で動作させるためには，OS やトランスポート層への依存が少ないほうが望ましい．

そこで，我々はアプリケーション（AP）間の通信を中継する仮想ネットワークとして，シームレスプロキシ（S-proxy と呼ぶ）で構成されるプラットフォームを構築することで，シームレスにサービスを連続させる技術を提案する[1),2)]．

S-proxy は送受信する通信データを保持するバッファおよび AP や AP プロトコル（APP）の状態を保存する状態キャッシュを持っているため，S-proxy 間の通信環境を隠蔽し，一時的な切断や他の接続手段への変更等通信環境の変化によるデータの欠損を防ぐ．さらに，S-proxy 間のつながり（経路）を変化させることで，ネットワークトポロジの動的な変化に対応し，状態キャッシュの情報を他の端末に転送し復元することで，ユーザの端末の変更に対応する．

シームレスサービスプラットフォームは 4 つの階層（S-Session，S-Connection，S-Path，S-Link）にモデル化され，これらの 4 つの階層間の関係を変化させることで，ノードの移動（node migration），中継経路の変更（relay route change），サービスの移動（service migration）といった 3 種類の切替えを実現する．このように我々の仮想ネットワークモデルは，ノードやユーザの自由な移動にともなう変化に適応し，シームレスにサービスを連続させることができる．

本論文ではシームレスなサービス連続技術をサポートするプラットフォームの実現方式について述べるとともに，実装したプロトタイプの評価結果を報告する．このプラットフォームにより，動的なネットワークの変化に適応し，ユーザに追従する究極のモバイルシステムが実現できる．

2. 要求条件

本論文で提案するシームレスサービスプラットフォームに対する要求条件は以下のようなものである．

(1) サービス連続技術をさまざまなアプリケーションプロトコル（APP）に適用できる．
(2) 既存の OS，アプリケーション（AP）への依存度が低い．
(3) ノードはアドレス☆の変更（ノードの移動）において，シームレスにサービスを連続させる．
(4) オーバレイネットワーク上の中継機能の変更（中継経路の変更）機能を利用し，両端の移動ノード（エンドノード）が移動した場合の共通の接続点を確保する．
(5) オーバレイネットワーク上の中継機能の変更（中継経路の変更）機能を利用し，ファイアウォール等で分断されているため，直接下位のトランスポート層で通信不可能である場合においてもオーバレイネットワーク上で接続可能にする．
(6) クライアントノードの変更（サービスの移動）において，シームレスにサービスを連続させる．

3. 関連研究

ここでは関連研究の位置づけを明らかにすることで，本提案の特徴を明らかにする．近年，動的なネットワークの変化に適応するシステムとして，インフラとなる通信設備を想定しないアドホックネットワークの通信プロトコルや，特定のサーバノードに依存しない P2P サービスが注目されている．しかしながら，MANET[3)] で考案されているアドホックネットワークにおいては，ルーティング（接続性）のみを考慮しており，無線，有線のネットワークが混在したり，既存バックボーンネットワークと連携したりするような

☆ さまざまなトランスポート層を利用するためにはノードはアドレスとしてプロトコルの種類とそのソケットの情報が必要である．たとえば TCP，IP アドレス，ポート番号の対等である．このとき，TCP を利用しポートは固定とすると，ネットワークアドレス（IP アドレス）と同義になるため，本論文ではそのように簡素化して扱う．

場合に，ノードの移動やサービスの移動を実現することは想定されていない．

P2P サービスにおいては特定のサーバに依存しないため，ネットワークトポロジの動的な変化の影響を受けにくい．それゆえ，動的なネットワークを扱うのに適している．実際，Jxta[4] の P2P モデルは我々の提案するシームレスサービスプラットフォームのモデルを包含していると見ることもできる．しかしながら，Jxta では自律的に通信相手ノードを発見することに重点がおかれているため，ノードが自由にネットワーク内を移動する際に短時間で通信相手ノードとの通信を回復し，シームレスにサービスを連続することを直接解決するものではない．たとえば，Jxta ではピアの機能は基本的な通信機能に限定しており，ピア間の接続が変化した場合にサービスを連続させる機能に関しては Jxta のモデルの範囲外である．

また，Groove[5] のようなコラボレーション機能に特化したサービスはスケジュール等のデータを同期させることにより，さまざまなノードから同じデータにアクセスすることでサービスの継続性を実現しているが，チャットの途中で瞬時に違うノードで再開するとサービスが中断する．このように通信中に通信そのものまで引き継ぐサービスの連続性までは実現していない．

我々は以前からノード間で完結するサービス連続性技術（MEDLAR）を提案してきた[6),7)]．S-proxy を用いシームレスな環境を構築する点では本提案と共通であるが，エンドノード間で直接通信するため，中継を行う S-proxy の機能がなく，エンドノードが自由に動き回った場合，再接続点がなくなり通信が継続できない．また，片方向に流れるストリームのみを対象としている．

IPMSA[8] は個人の設定環境や動作環境をあるコンピュータから別のコンピュータへ引き継ぐことができる．これは我々の提案と似ているが，通信中に通信そのものまで引き継ぐことはできない．

Mobile IP[9]，Mobile Session IP[10]，Seamoby，HAWAII[11]，Cellular IP[12]，TIMIP[13]，Migrate[14] は TCP/IP トランスポート層，TCP/IP インターネット層で実現するサービス連続性技術である．TCP/IP での実現は TCP/IP を利用する AP に影響を与えず，カーネルモードで実行されるため，パフォーマンス的にも有利である．そのため，シームレスなサービス連続技術には有用である．しかしながら，以下の理由によりユーザの要求に十分に応えることはできない．

- TCP/IP によるアプローチでは AP やノードのリソース情報（AP の状態やディスプレイのサイズ等）を直接扱うことができない．それゆえ，ノード間をサービスが移動するような場合，移動先のノードに AP の状態を転送し，AP の状態を再設定する機能や移動先のノードの能力に応じて，コンテンツを間引いたり変換（トランスコーディング）したりする機能をアプリケーションレベルで実現する必要がある．つまり，我々の提案するような AP や APP を意識した機能と連係させなくてはならない．そのため，TCP/IP で完結するといった利点は薄れる．
- ホームエージェント等の固定ノードの存在を想定しているため，ノードの移動性が制限されている．我々の提案でも中継機能は存在するが，固定されている必要はなく，動的に変更可能である．
- TCP がトランスポート層としてつねに有効であるとは限らない．IP ネットワークがファイアウォールで分断されていれば，HTTP をトランスポート層として利用する必要がある．我々の提案はサービスの連続性を実現する部分をトランスポート層の上位で実現しているため，下位のトランスポート層の種類に依存しない．
- TCP/IP によるアプローチでは，OS ごとに実装を行う必要があり，OS 内部のネットワーク機能の改良をともなう．これは多種多様な端末が存在するユビキタスなネットワーク環境では大きな障害となる．これらに比べて我々のシームレスプラットフォームは OS の実装と分離されているため，JAVA 等の OS 依存の少ない言語を用いることができ，多様な環境での動作が容易に実現できる．

4. シームレスサービスプラットフォームの構成

4.1 シームレスサービスシナリオ

図 1 にシームレスサービスプラットフォームの利用例を示す．

アドホックネットワーク，ホットスポット，自宅の LAN 環境で構築されるネットワークにおいて，以下のようなシナリオを想定する．

(1) 自宅の LAN 環境のノードとアドホックネットワーク内のノードで通信開始．
(2) アドホックネットワーク上のノードが無線 LAN のホットスポットに移動しネットワークアドレスが変更（ノードの移動）．
(3) 利用ノードを自宅のデスクトップパソコンから携帯性に優れた PDA 等に変更（サービスの

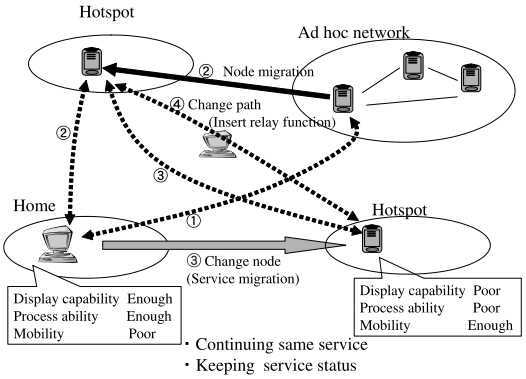

図 1 シームレスサービスシナリオ
Fig. 1 Seamless service scenario.

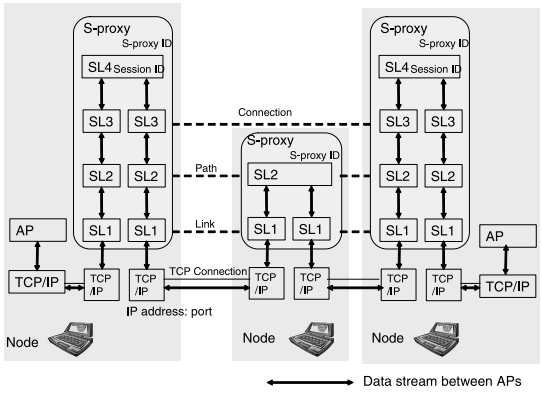

図 3 シームレスサービスのレイヤ構成
Fig. 3 Seamless service layer.

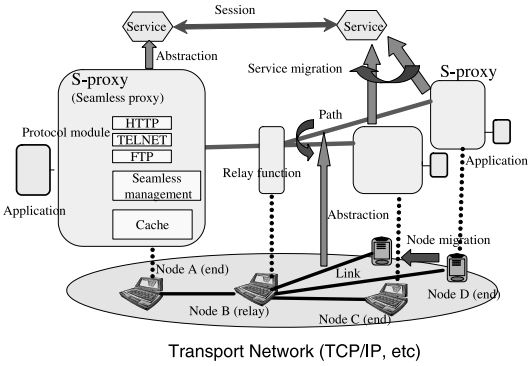

図 2 シームレスプロキシネットワークモデル
Fig. 2 Seamless proxy network model.

移動).

(4) 両方の端末が同時に移動することが想定されるため，間に安定的に中継するノードを挿入（中継経路の変更).

このように，シームレスサービスプラットフォームはネットワークの変化やユーザの要求が生じた場合にも，サービスの状態を保持して．同一サービスを中断点から再開する．

4.2 シームレスプロキシネットワーク

要求条件 (2) を満たすためには，OS や AP の機能とシームレスサービスを実現する機能（S-proxy）を分離する必要がある．そこで，OS の上位に複数の S-proxy で構成されるオーバレイネットワーク（S-proxy ネットワーク）を構築し，AP 間はこのオーバレイネットワークを介して通信しあう．

S-proxy の構成概要とそのネットワークモデルを**図 2** に示す．各ノードには S-proxy が配置されている．データの送受信は AP 間で行われ，AP 間で送受信されるデータは複数の S-proxy が中継している．

シームレスサービスにおける通信はサービスとサービスの間でやりとりされるデータの送受信を示し，個々の通信はセッションという単位で管理される．サービスとはユーザに同種または同一のコンテンツを配布する等の要求に応答する機能や，その要求を出したり，その要求に対する応答を表示，操作したりする機能を示す．サービスはノード上の AP によって実現される．ただし，同一のサービスは異る AP でも実現できる．

S-proxy 間ではシームレスプロキシプロトコル（SPP）により制御情報，通信データの送受が行われる．S-proxy はユーザや外部プログラムからの指示を受けたり，S-proxy 内部の状態が変化したりすると，SPP を用いて S-proxy 間や S-proxy-AP 間の通信の切断，再開等を行い S-proxy 間の接続関係が変化する．これにより，AP 間の個々の通信に対し，自由にネットワークやノードの構成を変えることが可能となる．

4.3 レイヤ構成

シームレスサービスプラットフォームのレイヤ構成を説明する．**図 3** にレイヤの構成と AP 間のデータの流れを示す．また，**図 4** に S-proxy 内部の詳細な動作を示す．OS で提供される TCP/IP トランスポート層の上位に 4 階層のシームレスサービスレイヤがある．各階層の動作はそれぞれの Manger スレッドが独立に制御している（図 4 の S-Session manger 等）．AP に直接接続しているエンドエンドの S-proxy 間は S-Connection（SL3）層のコネクションという概念で接続され，S-proxy 間は S-Path（SL2）層のパスという概念で接続されている．また，下位のトランスポート層を抽象化したのが S-Link（SL1）層のリンクである．ノードの移動，サービスの移動，中継経路の変更の 3 種類の切替えは各レイヤ間の対応関係の変化に相当する．

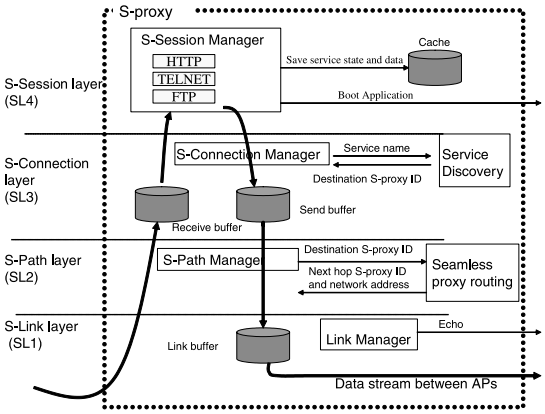

図 4 シームレスプロキシの内部処理
Fig. 4 Seamless proxy.

S-Session 層ではサービスどうしを結び付ける．サービスの結び付きのことをセッションといい，それぞれのセッションはセッション ID により識別される．ただし，セッション ID をそのセッションにかかわるすべての S-proxy で一意に設定することは複雑な手順が必要であるため，各 S-proxy を一意に識別する S-proxy ID とその S-proxy 内で処理しているセッションを一意に識別できる ID とを組み合わせたものにする．そのため同一セッションでも S-proxy によってセッション ID は異なるが，セッション ID を指定すればセッションを一意に特定できる．

S-Session（SL4）層では APP の状態を保存する状態キャッシュ（Cache）がある．S-Session 層では後述の S-Connection（SL3）層の受信バッファから通信データを取り出し，S-Connection 層の送信バッファに送る．この際，通信データの内容をチェックし，通信の状態，AP の状態等を状態キャッシュに保存する．また，S-Session 層には受信するノードの能力に応じて通信データを変換するトランスコーディング機能を実装できる．状態キャッシュへの保存，トランスコーディング機能は APP に依存した機能であるため，APP ごとに実装する．APP ごとの実装は S-Connection 層のバッファから読み込んだ情報をどのように処理して，状態キャッシュに何を保存するか，また状態キャッシュからどのように解釈して状態を復元するかを記述する．つまり，APP の違いは S-Session 層の処理方法の違いであり，各 APP ごとの機能モジュールを S-Session 層に加えれば各 APP に対応することが可能となる．状態キャッシュやその情報の転送等は共通化されており，トランスコーディング機能等，解析機能は既存のプログラムを転用することができるため，これらの記述は簡易なものとなる．つまり，他のレイヤの動作や S-Session 層内の多くの部分は共通化されている．

S-Connection（SL3）層ではエンドエンドの AP 間の関係を扱う．AP に直接接続されている S-proxy（エンド S-proxy）間の接続関係をコネクションと呼ぶ．AP はサービスを実現するためにノード上で動作しているソフトウェアを表す．たとえば HTTP でいう HTTPD や Web ブラウザといったものに相当する．S-Session 層のサービスは S-Connection 層のエンド S-proxy にマッピングされる．S-Session manager はサービス発見機能（Service discovery）にサービス名（Service name）で検索することにより，相手のエンド S-proxy ID（Destination S-proxy ID）を得ることができ，これを S-Path 層に通知する．S-Connection 層では通信データをバッファリングするともにエンド S-proxy 間で送達確認を行う．バッファには送信バッファ（Send buffer）と受信バッファ（Receive buffer）がある．送信バッファは送達確認がされるとそのデータが消去される．受信バッファには通信データとともに S-proxy 間の制御メッセージも蓄積される．通信データは FIFO で処理されるが，制御メッセージは優先的に先頭におかれ，即座に処理される．

S-Path（SL2）層は S-proxy どうしの接続を扱う．S-Connection 層から通知されたエンド S-proxy ID をキーに S-proxy ルーチング機能（S-proxy routing）に問い合わせ，エンド S-proxy までの中継 S-proxy ID とそのネットワークアドレスを取得する（接続経路情報）．また次に接続すべき S-proxy（次ホップの S-proxy）への接続を S-Link 層に指示し，リンクが確立したら，次ホップの S-proxy に接続経路情報を送信する．S-proxy 間のつながりをパスと呼ぶ．S-proxy は S-proxy ID によって特定されるため，パスは S-proxy ID の対によって表現される．S-Path 層は，S-Link 層にリンクの確立を指示し，S-Link 層でリンクが確立すると接続先の S-proxy に Create の制御メッセージを送信し，接続先の S-proxy からの応答として Create OK の制御メッセージを受信することによりパスが確立する．また，接続先の S-proxy に Remove 制御メッセージを送信し，接続先の S-proxy から Remove OK の制御メッセージを受信することによりパスが解消される．

S-Link（SL1）層では OS の提供する TCP/IP トランスポート層等に 1 対 1 で対応するリンクを扱う．S-proxy 間のリンク（TCP 等のトランスポート層のコネクション）は S-Path 層のパスに対応づけられる．S-Path 層からの指示により，次ホップの S-proxy までのリンクを確立する．次ホップの S-proxy に接続す

ると接続先の S-proxy と SYN 制御メッセージを発行しあい，リンクを確立する．S-Link 層ではその後リンクがつねにつながっているかを確認するため，定期的にリンクの確立した S-proxy 間で ECHO 制御メッセージを投げあう．

このように 4 階層のレイヤ構造をとることにより，S-proxy の機能を分割し必要な機能のみを利用することが可能となる．たとえば，図 3 の中継 S-proxy のように SL2 層までの機能を利用し単純にデータを中継する S-proxy や SL3 までの機能を利用し，送達確認まで行う中継 S-proxy を構築することができる．

4.4 サービス発見機能と S-proxy 間ルーチング機能

サービス発見機能はサービス名から宛先のエンド S-proxy ID を求める．サービス名はたとえば URL のようなもので表現できる．たとえば，http://192.168.0.1/test から S-proxy ID(0041) を求める．これはルックアップサーバのような集中的なテーブルで管理することもできるし，DHT[15] を利用し分散的に管理することもできる．

S-proxy ルーチング機能は S-proxy で構成される経路を構築するのに用いられる．各 S-proxy 間はネットワーク層のブロードキャスト等で周辺の S-proxy を発見したり，設定ファイル，コマンド等で，指定された S-proxy を認識したりしていく．お互いを認識している S-proxy 間では S-proxy ID とネットワークアドレスを交換し，S-proxy 間のルーチングテーブルを作成していく．一般的な TCP/IP インターネット層以下のルーチングテーブルと異なる点は S-proxy ID とネットワークアドレスの両方を同時に扱わなくてはならない点である．たとえば，ネットワークアドレス A で発信するときは S-proxy 41 を中継とし，ネットワークアドレス B で発信するときは S-proxy 31 を中継とする等，ネットワークアドレスにより中継を行う最適な S-proxy が異なることが考えられるからである．

このような機能は本システムにとって重要な機能であるが，その実現方法は既存の技術を利用できる部分が多く，また，オーバレイネットワークの規模，サービスの用途によっても変化してくる．たとえば，サービス発見に Jxta，S-proxy 間ルーチング機能にソースルーチングを用いることが考えられる[17)~20)]．本論文ではこれらの機能を利用することを前提に議論する．

また，S-proxy ID の一意性を保証する技術については MAC アドレスから算出する等適切なものを利用することとし，本論文では一意性は保証されるものと仮定し議論する．

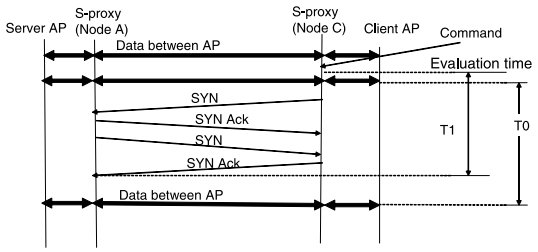

図 5 ノード移動のシーケンス
Fig. 5 Sequence of node migration.

4.5 ノードの移動（Node Migration）

ノードの移動は S-proxy に対するネットワークアドレスの変更とみることができる．ネットワークアドレスの変更はノードの位置が物理的に移動して，通信デバイスの接続アドレスが変化する場合とノードに接続されている複数の通信デバイスを選択しなおした場合に起こる．このとき，S-proxy ID とネットワークアドレスの対応関係が変化する．これは S-Path 層のパスと S-Link 層のリンクの対応関係が変化したと表現できる．S-Link 層の ECHO 制御メッセージの不到達により，リンク断を検出するか，ユーザや外部プログラムが発行したコマンドにより，ネットワークアドレスの変更を命令されると S-Path 層は経路の再構築を行い，S-Link 層に次ホップの S-proxy へ再接続を指示する．

図 5 にノード移動の例を示す．図の $T0$，$T1$ は評価の節で扱うこととする．ここではノード C の S-proxy が外部からネットワークアドレスの変更を命令され，ノード C の S-proxy とノード A の S-proxy 間でリンクを張りなおし，S-proxy 間のパスに再マッピングしている．結果としてノード C の S-proxy は新しいネットワークアドレスで通信を継続する．S-Link 層は新しいリンクができると S-Path 層に通知し，S-Path 層で既存のパスと新しいリンクを対応づける．S-Connection 層でエンド S-proxy 間の送達確認と再送処理が行われるので，下位の S-Link 層で新しいリンクに変化した場合でも，データの欠損は起こらない．

4.6 サービスの移動（AP 切替え）

あるノードの AP を異なるノードの AP に変更しサービスを連続させることをサービスの移動（パーソナル/セッションモビリティ）という[16)]．たとえば，PDA で動作していた AP からパソコンで動作する AP へ変更し，サービスを継続させるといった場合が考えられる．

図 6 はノード B の AP からノード C の AP に変更した例を示している．このように，コネクションの端点

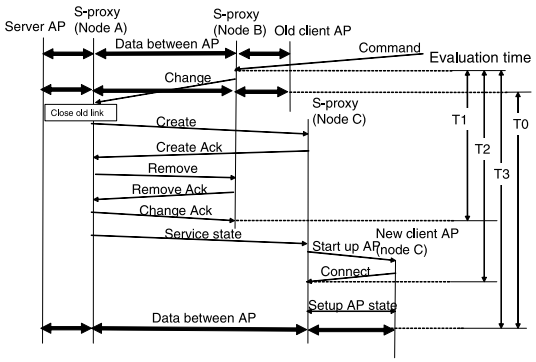

図 6 サービス移動のシーケンス
Fig. 6 Sequence of service migration.

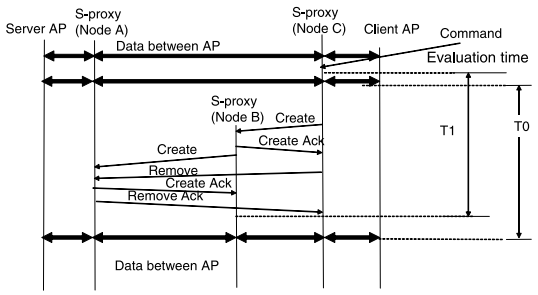

図 7 中継シームレスプロキシ変更のシーケンス
Fig. 7 Sequence of relay S-proxy change.

をたとえば PDA とパソコン間といった具合に自由に移動し，その場合にも，同一セッションが維持できる．これは S-Session 層のセッションと S-Connection 層のコネクションとの関係を変化させることに相当する．

S-Path 層では，Create 制御メッセージによりノード A の S-proxy とノード C の S-proxy 間のパスを構築し，Remove 制御メッセージによりノード A の S-proxy とノード B の S-proxy 間のパスを解消する．S-Session 層では，ノード A の S-proxy の状態キャッシュに蓄えられたサービス状態がノード C の S-proxy の状態キャッシュに転送される．さらに，この状態キャッシュの情報をもとに，ノード C 上で AP を自動的に起動する．既存の AP に対してこの状態を設定する方法としては

(1) 起動コマンドの引数とする，
(2) AP が公開している API を利用する，
(3) expect 等の自動入力スクリプトを利用する（Telnet FTP のログイン，作業ディレクトリの変更等），
(4) AP に状態を設定しないで，AP から送信された通信内容を S-proxy 上の S-Session 層で変更し，同等の動作を行うことにより模倣する（Web の Cookie 等），

等が考えられる．起動のための簡単なスクリプトを記述する必要があるが，多くの既存 AP で AP の状態を他のノードの AP に自動設定することができる．

4.7 中継経路の変更（S-proxy 切替え）

中継 S-proxy を追加，削除，または他の中継 S-proxy に変更できる．変更の契機としてはルーチングテーブルの変更のような S-proxy 内部からの指示と外部プログラムやユーザによるコマンド指示によるものがある．

図 7 に中継経路の変更例として中継 S-proxy（ノード B）を S-proxy（ノード A）と S-proxy（ノード C）の間に挿入する場合のシーケンスを示す．これは S-Path 層のパスと S-Connection 層のコネクションとの関係を変化させることに相当する．S-Path 層において Create 制御メッセージによりノード C の S-proxy とノード B の S-proxy 間，ノード B の S-proxy とノード A の S-proxy 間のパスを構築する．また，Remove 制御メッセージによりノード A の S-proxy とノード C の S-proxy 間のパスが解消される．

S-proxy に中継機能を持たせることにより，以下のような動作が可能となる．

(1) 両端の移動ノードが同時に移動した場合でも，共通の接続点とすることができる．図 5 で説明したノードの移動はパスで接続された S-proxy 間で解決できるため，S-proxy により中継させることで，両端のノードの移動シーケンスを独立に行うことができる．そのため，両端のノードが移動する場合でもお互いのノードを見失うことなく，安定的にノードの移動を実現できる．
(2) ノードの移動先ネットワークがファイアウォールや NAT 等で分断されている場合，ファイアウォール上に S-proxy をおき中継させる．これにより，直接 IP 到達不能なネットワーク間でも本技術を適用することが可能となる．また，ファイアウォール上に S-proxy の設置が困難であるようなセキュリティの制約の強い環境でも，下位のトランスポート層として HTTP を利用し，S-proxy で中継する際に HTTP を終端させ，TCP/IP に変換することも可能である．

5. 実装と評価

シームレスサービスプラットフォームのプロトタイプとその評価を報告する．プロトタイプの S-proxy は Java で作成し，Red Hat7.2 上の AP に対応している．この S-proxy は提案している 3 種類の切替えを実現できる．

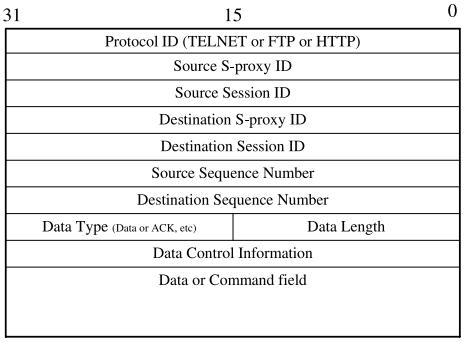

図 8 シームレスプロキシプロトコルにおけるパケット
Fig. 8 Packet on seamless proxy protocol.

また，中継 S-proxy も AP と接続するエンド S-proxy も同じプログラム，プロセスで動作し，TCP の接続ポートの違いにより，動作を切り替えている．

今回は S-proxy 間の接続関係を決定するときに利用する S-Path 層のルーチングテーブルは静的なものに限定した．このルーチングテーブルは接続可能な S-proxy どうしの S-proxy ID の対とその IP アドレスが，すべて記載されている．そして，これを参照することにより，次に接続すべき S-proxy を決定する．また，複数の経路候補から経路を選択するための評価パラメータとしてコスト値を設定しておく．ユーザのコマンドによって中継経路が指定された場合はユーザのコマンド指示を優先する．また，静的なルーチングテーブルのため，ルーチングテーブルの変更にともなう，中継経路の変更は起きない．

また，サービス発見機能はサービス名を URL および IP アドレスにし，この IP アドレスをキーにルーチングテーブルの情報から S-proxy ID を得ることによって代用した．

S-proxy の S-Connection 層間でやりとりされるパケットを**図 8** に示す．このパケットはヘッダと可変長のデータから構成される．シーケンスナンバは S-Connection 層で付与され，各パケットを区別し送達確認に用いられる．そのほかにも各 S-proxy 間で S-Path 層のやりとりに用いられる Create，Remove 等の制御メッセージ用のパケットと S-Link 層でのやりとりに用いられる SYN，ECHO 等の制御メッセージ用のパケットがある．

また，既存のソフトウェアを AP として利用し，AP の起動と状態設定をするラッパプログラムと S-Session 層に各 APP 固有の動作を記述するモジュールを追加することにより，さまざまな APP に対応した．

次に APP をステートフルとステートレスに分類し，それぞれの特徴にあったサービス移動の実現方法を述べる[16]．

5.1 ステートフル（対話型）APP におけるサービスの移動の実現

遠隔の端末とステートフルで対話型の通信を行う APP として，TELNET，FTP，電子会議等に用いられる SIP 等がある．

これらのプロトコルの特徴は通信相手との通信シーケンスにより AP の状態が遷移することにある．そこで，サービス移動を行う際，S-proxy が移動先で起動したクライアント AP と切り替える前までに行った状態遷移にかかわる通信シーケンスを，切替え先のクライアント AP 起動後にクライアント AP から実行させる必要がある．また，サーバ AP はすでに旧クライアント AP と上記通信シーケンスを実行済みなため，切替え先のクライアント AP に対して適切に応答することができない．そのため，クライアント AP 側のエンド S-proxy がサーバ AP のかわりにクライアント AP との通信シーケンスの相手をすることにより，状態を遷移させる必要がある．

ここでは，本実装で対応している Telnet，FTP を例に実現方法を述べる．

Telnet，FTP におけるセッションの単位は，Open コマンドで作られる通信間ではお互いのセッションの状態が干渉しないので，Open コマンドから Close コマンドまでとする．

サービス移動先のノードで S-proxy が切替え時の通信シーケンスを実行するように指定し起動したクライアント AP は通常サーバ AP と最初に行うログイン等のネゴシエーションの通信シーケンスを実行する．このとき，サーバ AP にクライアント AP の切替えを隠蔽するため，クライアント AP 側のエンド S-proxy の S-Session 層はサーバ AP のかわりに上記通信シーケンスに対し応答する（サーバ AP はサービス移動前のクライアント AP とネゴシエーションの通信シーケンスが完了している）．また，ユーザインタフェースの観点からそれまでの画面の数十行程度を移動先に表示する等の状態の継承を行うこともできる．

FTP では Telnet と異なり制御用コネクションとデータ転送用コネクションが別々に構成される．そこで，データ転送用コネクションを再現するために，サービス移動先のクライアント AP を S-proxy から起動する際，クライアント AP がデータ転送用コネクション作成コマンド（PORT コマンド）を発行するようにする．その後，サーバ AP 側のエンド S-proxy は S-Session 層でそのコマンドを読み取り，サーバ AP との間で保持しているデータ転送用コネクションと接

続する．また，作業ディレクトリや転送されるデータ形式である TYPE（ASCII，EBCDIC，IMAGE）の情報を新クライアント AP で引き継ぐ等の S-proxy の S-Session 層で状態キャッシュの状態を新クライアント AP に設定することで，新旧のクライアント間で状態の継承を行う．

5.2 ステートレス（Request-Response 型）APP におけるサービスの移動の実現

ステートレス APP として HTTP がある．このプロトコルの特徴は通信相手とのシーケンスは 1 リクエストにつき 1 レスポンスといった小さな単位で完結することである．そのため，通信シーケンスにより AP の状態を遷移させる必要はなく，AP の状態は単にデータとして保持すれば十分である．そこで，サービス移動を実現する手順は

(1) サーバ AP と直接接続している S-proxy の S-Session 層の状態キャッシュの情報を移動先のクライアント AP と直接接続する S-proxy の状態キャッシュに転送する，
(2) 移動先のクライアント AP の存在するノードで S-proxy はクライアント AP を起動する，
(3) クライアント AP から直接接続された S-proxy の S-Session 層はクライアント AP のリクエストを状態キャッシュの情報をもとに変更を加え，サーバ AP 側のエンド S-proxy に向けて送信する，

となる．

ここでは，本実装で対応している HTTP を例に実現方法を述べる．HTTP の利用方法には大きく分けて 2 種類ある．動画等比較的データ量が大きい場合と，HTML の文章，静止画等比較的データ量が少なく，リクエストとレスポンスがすぐに終了してしまう場合である．1 つのセッションが終了してしまうと保持した状態キャッシュがクリアされてしまうため，それぞれに対し，以下のように別の方式を採用した．

- 1 リクエストを 1 セッションに対応させる方式
  動画の表示等，1 リクエスト，1 レスポンスで完結し，かつデータ量が多く通信時間が長い場合に適している．移動先の S-proxy ブラウザを起動する際，URL にセッション ID を含めることで，どのセッションであるかを識別する．
- 複数リクエストを 1 セッションに対応させる方式
  HTML では複数の文章ファイルと複数の画像ファイルで 1 つの画面を構成するため，1 つの画面を表示するためには複数のリクエストとレスポンスで状態を保持する必要がある．また，オンラインショッピング等，Cookie を用いて，複数のページを 1 つのセッションのように扱うこともある．本方式はこのような場合に適している．実際にはユーザがハイパーリンクをたどっている間を 1 セッションとする．移動先のクライアント AP でも同一の Cookie 情報を扱うため移動前から各 S-proxy の S-Session 層で Cookie 情報を状態キャッシュに保存し，これを移動先のクライアント AP 側のエンド S-proxy の S-Session 層の状態キャッシュに転送し，この S-proxy の S-Session 層で移動先のクライアントからのリクエストに Cookie 情報を付加する．また，S-Session 層は HTML の解析，変換機能を持つので，これを利用して複数のセッションを同時に扱うことができる．つまり，クライアント AP と直接接続される S-proxy はクライアント AP に HTML ファイルを転送する際，S-Session 層で HTML の SRC 属性，LOWSRC 属性，HREF 属性，ACTION 属性に含まれる URL にセッション ID をあらかじめ含める．その結果，ユーザがこれらのリンクをたどることによってクライアント AP から発行されるリクエストがどのセッションであるかを S-proxy で識別できる．

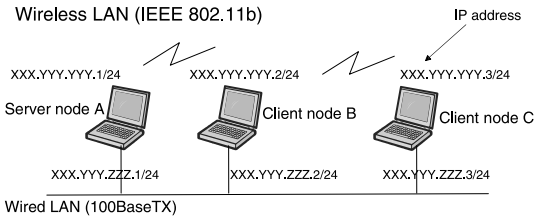

図 9 実験環境
Fig. 9 Experimental setup.

5.3 動 作 実 験

プロトタイプを用いて TELNET，FTP，HTTP で動作実験を行った．実験環境を**図 9** に示す．各ノードで S-proxy が動作し，ノード間は有線 LAN と無線 LAN で結ばれている．ルーチングテーブルでは無線 LAN より有線 LAN の経路が優先して選択されるようコストの値が与えられている．

動作実験は 4.1 節で説明した動作シナリオを模擬するため以下の手順で行った．

(1) ノード A とノード B 間で有線 LAN を使い通信を開始し，有線 LAN のケーブルを抜くことでリンク切替え（ノードの移動）が起こり，無線 LAN を介した通信に変更される．その後，有線 LAN のケーブルを差すことでリンク切替えが起こり，有線 LAN を介した通信に変更さ

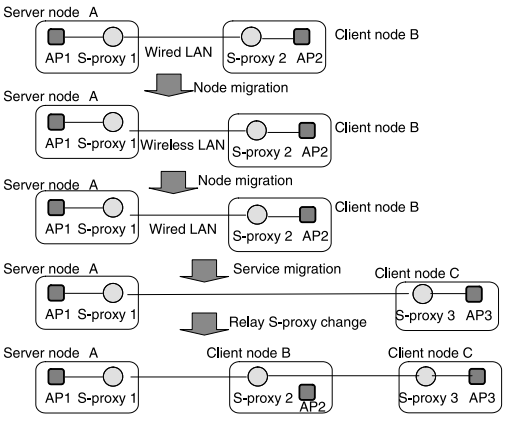

図 10 動作手順
Fig. 10 Experimental procedure.

れる．

(2) ユーザのコマンド投入によりノード B からノード C にクライアント AP を切り替え（サービスの移動）ノード C とノード A 間を有線 LAN を介して通信する．

(3) ユーザのコマンド投入により指定した S-proxy を中継経路とする，S-proxy（ノード C）と S-proxy（ノード A）間に S-proxy（ノード B）を挟んだパスを生成する（中継経路変更）．

この動作実験における S-proxy 間の接続関係の変化を**図 10** に示す．

切替えコマンドはどのノードからでも発行できるが，この実験ではクライアント AP が動作しているノードで発行した．

S-Link 層における接続確認の ECHO は 1 秒間隔である．この ECHO を用いることにより，S-proxy はデータ送受信中でなくても S-proxy 間のリンク断を検出することができ，自動的にリンク切替えを行うことができる（ユーザや外部プログラムが発行するコマンドからリンク切替えを行うことも可能である）．

ユーザが入力したコマンドは S-proxy のコマンド待受けのための TCP ポートに送られるため，このポートで受信した時点をコマンド投入時間としている．

ノードの移動，サービスの移動，中継経路の変更の 3 種類の切替えに関し切替え時間を tcpdump によって測定した．3 つのノードの時刻は NTP によって同期している．NTP におけるノード間の誤差は 1 ミリ秒以下である．

5.4 切替え時間の評価

3 つの切替えの動作シーケンスと測定時間は図 5，6，7 と同様である．

APP は Telnet，FTP，HTTP を用いた（FTP ではファイルのダウンロード中に切替えが行われる）．(1)，(3) においては，HTTP は Telnet と同じような動作をすると考えられるため省略した．

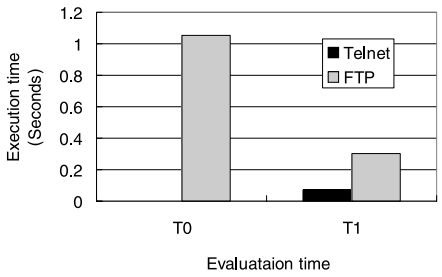

図 11 ノード移動の切替え時間
Fig. 11 Evaluation time for node migration.

5.4.1 ノードの移動

図 5 の $T0$，$T1$ を測定した．$T0$，$T1$ は ECHO によるリンク断を検出するための時間は含まれず，検出されてからの時間を測定している．$T0$ はサーバ AP の存在するノードとクライアント AP の存在するノード間の通信の切断時間に相当する．$T0$ は通信データの途切れる受信間隔を測定するため，切替え前後でとぎれなく通信データが S-proxy 間でやりとりされている必要がある．このためこのようなデータの送受信が容易な FTP のみ測定した．$T1$ は S-proxy 間の処理時間で，切替えコマンドを受けてから，実際に S-proxy が切り替わるまでの時間である．

切替え時間の測定結果を**図 11** に示す．

5.4.2 サービスの移動

図 6 の $T0$，$T1$，$T2$，$T3$ を測定した．

$T0$，$T1$ の定義はノードの移動と同様である（$T0$ は FTP のみ測定した）．$T2$ は切替えコマンドを受けてから切替え先のノードで AP が起動するまでの時間である．$T3$ は切替えコマンドを受けてから，AP の状態を切替え先の AP で引き継ぎ，切替えを完全に完了するまでの時間である．

サービス移動における HTTP は 5.2 節の複数リクエストを 1 セッションに対応させる方式を用い，HTML のファイル（2.9 KB）と 2 つのイメージファイル（2.3 KB および 1.2 KB）を含むページを表示している際に切替えを実行した．

切替え時間の測定結果を**図 12** に示す．切断時間 $T0$ は 1.5 秒で済んでいる．Telnet と HTTP に関しては $T1$ は 0.6 秒以内であり，$T2$，$T3$ は AP に依存している．HTTP において $T2$ と $T3$ は Telnet と比べてかなり大きい．これは Web ブラウザ（Mozilla）は Telnet のコマンドラインプログラムに比べてかなり容量の大きいプログラムなので起動に時間がかかるから

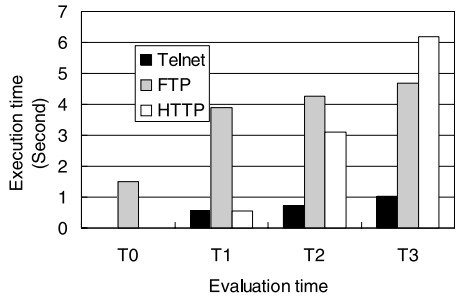

図 12 サービス移動の切替え時間
Fig. 12 Evaluation time for service migration.

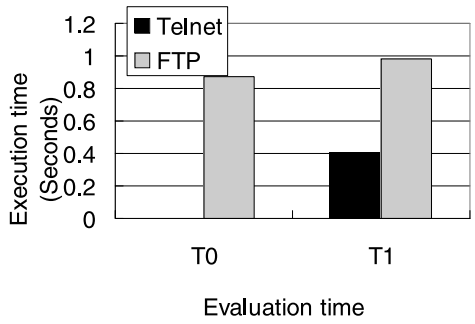

図 13 中継シームレスプロキシの切替え時間
Fig. 13 Evaluation time for relay S-proxy change.

である．

5.4.3 中継経路変更

図 7 の $T0$，$T1$ を測定した．$T0$，$T1$ の定義はノードの移動と同様である．切替え時間の測定結果を**図 13** に示す．

5.4.4 全体の分析

すべての切替えで $T1$ は Telnet では 0.5 秒以内であるが，FTP ではもっと大きい．これは FTP においてはファイルのダウンロード中に切替えが実行され，カーネルプロセス，ネットワークカード内のバッファに通信データが蓄えられている状態から，それまでに利用していた古いリンクを開放しなくてはならないためである．Telnet は軽い AP で起動時間が少なく，通信が断続的かつ小量なため古いリンク開放が瞬時に行える．したがって，AP の起動，古いリンクの開放を含まなければ，切替え時間は Telnet の $T1$ 以下であると考えられ非常に短い．サービスの移動において，FTP の $T1$ は 4 秒とノードの移動，中継経路の変更における FTP と比較して長い時間がかかる．これはサービスの移動の場合はサーバ AP が動作しているサーバノードから古いパスを切断しているため，クライアントノードに投入された切替えコマンドをファイルのダウンロード中にサーバノードに伝えるのに時間

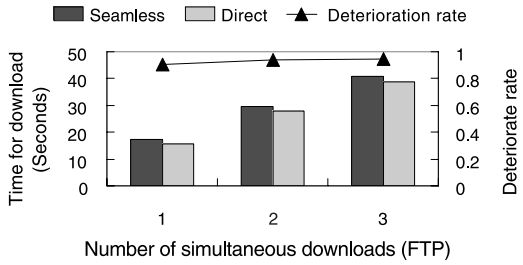

図 14 転送時間
Fig. 14 Overhead on S-proxy.

がかかるからである．しかしながら，この場合も切断時間 $T0$ は 1.5 秒で済んでいる．

また，切断時間 $T0$ はすべての場合において 1.5 秒以内である．この切断時間は TCP のハンドオーバ技術[14] における切断時間 1 秒に比べてもそれほど長い時間ではない☆．なぜならノードの移動と中継経路の変更に関してはほぼ 1 秒であるし，サービスの移動においては AP を起動する時間まで含まれているからである．それゆえ，本提案は十分に実用的であるといえる．

5.5 S-proxy を用いることによるオーバヘッド

S-proxy を利用することによる転送のオーバヘッドを測定するため，ファイルのダウンロード時間を測定した．有線 LAN を用い，FTP で 12.725 MB のファイルを以下の 2 通りの方法でダウンロードした．

- 2 つの S-proxy を介してダウンロード
- 直接ダウンロード

図 14 より S-proxy を利用すると，直接ダウンロードする場合に比べて，5〜10%多くダウンロード時間がかかる．これは我々が提案している S-proxy のオーバヘッドに相当する．このオーバヘッドの要因には，S-proxy 間の SPP パケットのヘッダの送受信や S-proxy 間の送達確認，S-proxy 上の処理遅延等が考えられる．

また，図 14 から同時にダウンロードをするファイルの数を増やしていくと，オーバヘッドが少なくなるということが分かる．つまり，転送時間の増加に対して，転送時間の差は 1.7〜2.1 秒とそれほど変わらない．これには 2 つの原因があると考える．1 つの原因

☆ 帯域が異なるため単純には比較できないが，ほとんどの本方式を含むほとんどの切替え方式においてノードの移動における切替え処理の主要部分は切替え前の TCP コネクションで利用している通信デバイス等のバッファにあるデータを送りきる部分である．TCP は帯域によりフロー制御を行っているので，その帯域を目一杯利用するようなデータ通信においてはこれらのバッファ上のデータ量は帯域に対して一定の割合であると仮定でき，切替えにかかる処理時間はある程度均一化されていると仮定できる．

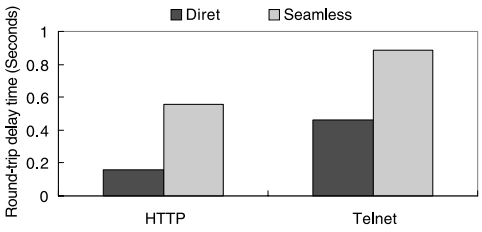

図 15 遅延時間
Fig. 15 Delay time on S-proxy.

はプロキシを中継することにより遅延が発生するが，同時にダウンロードするファイル数にかかわらず，その伝送路の帯域分のデータを転送をすることには変わりないため，処理遅延の影響は一定となる．したがって，全体の転送時間が長くなれば，その割合は小さくなる．もう 1 つの原因はプロキシ間で最初に行われるプロキシ間のネゴシエーションにかかる一定時間である．これも転送時間が長くなれば，その割合は減っていく．

また，HTTP，Telnet の場合のラウンドトリップの遅延時間を測定した結果を**図 15** に示す．クライアント AP 動作ノードで Tcpdump のタイムスタンプにより遅延時間を測定した．HTTP の場合はサービスの移動の場合と同じページを表示し，Telnet の場合は ls コマンドの結果を表示させた．測定結果から S-proxy を中継することにより，レスポンスが 2〜3.5 倍増加することが分かる．

今回は他の通信の干渉なしに遅延数ミリ秒程度の 100BASE-TX の LAN 環境で実験を行った．しかし，無線環境や広域なネットワーク環境を考えると遅延は数百 ms になり，他の通信の影響によりスループットも数倍という単位で変動する．したがって，S-proxy を用いるオーバヘッドが及ぼす影響の割合は減少していき，他のネットワークの要素にくらべ，突出した問題ではないと考えられる．

また，この評価実験において，通信路に対する SPP パケットの最適化等の処理はまったく行っておらず，現在詳細なログ収集等を行っている．したがって，SPP パケット長や送達確認の間隔等を調整することにより，転送時間の減少あるいは遅延の減少といった改善の余地は残されている．

5.6 コード量

表 1 に本実装の機能別コード量を示す．この表から APP 個別のコードやトランスポート層に依存しないコードは 600 kbyte で依存する部分はそれぞれ全体の 1 割前後であることが分かる．

表 1 コード量
Table 1 Quantity of code.

Layer	Class	Quantity of code(kbyte)
S-Session	Base	57
	Telnet	34
	Ftp	90
	Http	84
S-Connection		102
S-Path		17
S-Link	TCP/IP	51
Other	Proxy control, etc.	424

6. 考　　察

6.1 要求条件の充足

S-session 層の基底クラスから APP 独自の実装を行うクラスを派生させることで各 APP に対応可能である．この派生クラスには通信データを解析し，APP の状態，AP の状態を把握し，これを状態キャッシュに積む操作と，この状態キャッシュの情報から，AP の起動を行い，中断時の状態を復元する機能を記述することになる．プロトタイプの実装により，S-Session 層でも状態キャッシュの構成等共通部分があるほか，他の層や全体の制御等 8〜9 割のコードは共有することができることが実証された．TCP/IP より上位のミドルウェアとして実現することで，OS への依存部分をなくし，多くの OS で共通に利用可能である．このコードは Linux だけでなく MS-Windows 上でも動作確認をした．また，TCP/IP に依存する部分は S-Link 層に局所化され，今後他のトランスポートプロトコルの実装も容易であると推察される．4.6 節に示したように状態を復元する際の制約条件はあるものの，制約条件が緩いため，多くの既存 AP はこれをクリアしている．たとえば S-proxy が直接 AP に問い合わせて，AP の状態を得ることができなくても，S-proxy を通過する APP の内容をその S-proxy の S-Session 層で解析し，AP の状態を取得できる．また，取得した AP の状態を切替え先の AP に設定できなくても，切替え先の AP 側のエンド S-proxy の S-Session 層でそれを模擬することであたかも状態の設定がされているかのように振る舞うことができる．したがって，要求条件 (1)，(2) を満たしている．

S-proxy 間のパスは複数のリンク（トランスポート層のコネクション）を動的に変更させることが可能であるため，ネットワークアドレスの変更，通信デバイスの変更を S-proxy 間で隠蔽することが可能である．下位層に TCP/IP トランスポート層を利用した場合で

も S-Connection 層の送達確認でデータの連続性が保証されるため，TCP のコネクションを強制的に Close することがき TCP のコネクションの正常な終了を完了させる必要はない．したがって，要求条件 (3) を満たしている．

中継経路の変更は S-proxy 間のパスの構成を変更させることにより，実現している．S-proxy によって中継することにより，クライアント AP が動作しているノードとサーバ AP が動作しているノードは独立にノードの移動ができ，ノードの移動の安定性が増している．また，IP で到達できない分断されたネットワーク間を S-proxy により中継することができる．したがって，要求条件 (4)，(5) を満たしている

サービス移動は S-proxy 間のパスを切替え先のノードの S-proxy に変更し，S-proxy で保持していた AP や APP の状態を切替え先の AP に設定することで実現される．したがって，要求条件 (6) を満たしている．

このように，本提案は 2 章で述べた各要求条件を満たしている．

6.2 セキュリティ

分散システムにおいてセキュリティは重要な問題である．S-proxy 間の通信に Diff-Hellman 鍵交換方法[23] のような既存の公開鍵暗号技術を適用することは可能である．

しかしながら，セッション開始時のノードの認証は非常に困難である．なぜなら，動的なネットワーク環境において，信頼性が高く，いつも接続できる CA（認証局）を仮定することはできないからである．CA を分散させる技術[24] と CA 構築の効率性はトレードオフで，これが提案プラットフォームのセキュリティの制限となる．

セッション開始されると S-proxy 間でセッション ID が交換される．セッション開始後はこの交換されたセション ID と同じセッション ID で接続してくるノードのみ接続を受け付けることで，新たなノードが正当なものであるか認証する．これにより，第 3 のノードが不正にセッションを乗っ取ることは困難となる．そのためには，セッション開始時に各ノードの S-proxy 間でやりとりされるセッション ID を類推困難なものにしておく必要がある．また，サービスの移動の際はセッション ID を切替え先のノードになんらかの手段でセキュアかつユーザの手間なしに転送する必要がある．この手段としては，セキュアな IC カードをノードに差し込んだり，IrDA をもちいて，直接ノード間認証情報（セッション ID）を安全に交換することが考えられる．

6.3 本提案の可能性と限界

今回は中継 S-proxy の選択を静的なルーチングテーブルで行っている．そのため，今後は P2P 等の技術と連携し，動的なネットワークの変更を把握しそれに適応していく仕組みが必要である[17]．

原理的にはサービスの移動は多くの AP で行えるが，それが必ずしも，効果的であったり，適切に行えるかはそのサービスや AP の性質による．たとえば FTP のクライアント AP を他のノードに移動（サービス移動）する場合，カレントの作業ディレクトリ，転送モードを移動先のノードで引き継ぐことでユーザの操作によるそれまでの設定を継承するといった効果が期待できる．しかし，GET，PUT 等を途中から引き継いでも，転送ファイルが断片として分割されてしまったり，正しく 1 つのファイルにするために S-Session 層の状態キャッシュにファイルまるごと保持してそれを状態として転送（ファイル同期）するといった切替えにコストがかかる方法をとらなくてはならない．他方，ファイルの断片でも MPEG ファイルのように GOP 単位で再生可能なため利用価値のあるものもある．また，https，ssh といったエンドツーエンドで暗号化するようなプロトコルでは S-Session 層で APP の状態を解析するために，クライアント AP に接続している S-proxy で復号化するような機能を付与することが必要である．復号化する手段が通信シーケンスから類推できない場合，そのようなプロトコルを扱うことは困難となる．また，P2P やストリーミングのように APP の動作が公開されていなかったり，動作が複雑だったりする場合は，S-proxy で APP の状態解析を行う方法の非効率さが健在化してくる．

また，S-Session 層ではクライアント AP そのものの状態まで把握することはできない．そのため，HTML に埋め込まれたスクリプトやアプレット等通信シーケンスと無関係に変化するクライアント AP の内部状態までは扱うことができない．このようなアプリケーションプログラムが移動した場合の連続性に関してはモバイルエージェントのような別の分野の技術の範疇となり，そのような機能のない既存のアプリケーションで実現することは困難である．ただし，S-proxy からクライアント AP に対して内部状態についてアクセスすることができ，かつサービス移動先のクライアント AP に対して外部から内部状態を設定できる場合は，通信を解析することにより把握可能な状態と同様にクライアント AP の内部状態をサービス移動先のノードに転送し，再現することが可能である．

今回はトランスコーディングの機能は入れていな

い．しかし，S-Session 層は HTML の解析変換機能を有しているため，サーバ AP と直接接続している S-proxy において，Change 制御メッセージにクライアント AP の動作しているノードのリソース情報を含めることにより，PDA 用に HTML の高機能なタグを低機能なものに変換することも容易に実現できる．たたし，今回実装した S-proxy による中継は，S-Path 層までで処理しており，S-Session 層を介していないため，このままではトランスコーディングの機能を中継する S-proxy で実現することはできない．そこで，セッションとコネクションを一端終端し S-Session 層を通過するような中継機能を実現する等の改良が必要である．

PDA 等資源が少ないノードでは S-proxy の処理負荷の問題が健在化してくる．S-proxy を動作は，キャッシュを AP と重複して持つため，動作させない状況に対しておおよそ 2 倍の CPU，メモリの資源を消費することが予想される．また，今後，サービス発見と S-proxy 間のルーチング機能を加えることで，プログラムコードの量や処理負荷が増え，資源の消費が増大することが予想される．しかし，これらの追加機能は S-proxy 間で階層関係を構築できるため，資源に余裕のある中継 S-proxy にネットワーク全体把握を担わせ，資源の限られたノードの S-proxy は中継 S-proxy に接続する機能だけに限定できる．さらに，利用 APP を絞り込めばプログラムコードの量はおさえることができ，PDA で実現可能な範囲であると考えられる．しかし，携帯電話のようにさらに資源の限られた端末ではクライアント AP そのものを改良し，S-proxy の機能を簡素化し，両者一体として，構築する等汎用性を犠牲にしたほうが現実的である．

ノードの移動に関しては，アドホックネットワークのルーチング技術や IP レベルの移動技術で解決できる部分も多い．しかし，IP のネットワークは NAT やファイアウォール等で分断されているし，さらにこれらの技術は OS，ルータ等との依存関係が強いためつねに最適な解決方法とまではいいがたい．他方，我々のプラットフォームはトランスコーダの機能も組み込めるし，下位のトランスポートプロトコルに依存しない．したがって，本プラットフォームとネットワーク層等下位の移動技術を連係して動作させることにより，相互の長短所を補間しあうことができる．

今回はクライアントノードの変更のみを扱った．サーバのサービスを複数のノード上に複製（シャドウ化）し，同一サービスを複数のノードで提供すれば，サーバノードにも適用することが可能である[25]．

また，S-proxy で通信を分岐させれば，マルチキャスト通信等に容易に拡張することができる[26]．

7. ま と め

本論文では人間の移動，ネットワークの動的な変化にともなうネットワークの分断，融合およびユーザ環境の変化にともなう利用ノード，利用通信デバイスの変更に対し，シームレスにサービスを継続するシームレスサービスプラットフォームを提案した．また，我々の提案したモデルの 4 階層間の対応関係を変化させることにより，上記のような，環境の変化に適応できることを示した．本プラットフォームは既存のアプリケーション，アプリケーションプロトコル，OS 等を変更なしに利用できることを特徴としている．そこで，Telnet，FTP，HTTP といったアプリケーションプロトコルを対象にプロトタイプを作成し，既存のアプリケーションを用いてフィージビリティを確認した．また，評価実験において環境の変化に追従する切替え時間，プラットフォーム利用による転送速度，遅延時間を測定し，十分実用的であることを確認した．

参 考 文 献

1) 高杉耕一，田中 聡，中村元紀，久保田稔：動的なネットワーク環境上でユーザに追従するシームレスサービスプラットフォーム，DICOMO 2002, pp.297–300 (2002).
2) Takasugi, K., Nakamura, M., Tanaka, S. and Kubota, M.: Seamless Service Platform for Following a User's Movement in a Dynamic Network Environment, *1st IEEE International Conference on Pervasive Computing and Communications* (*PerCom2003*), Fort Worth, USA, pp.71–78 (2003).
3) Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, RFC2501 (1999).
4) draft-duigou-jxta-protocol, IETF draft rfc (2002).
5) Groove Network, Inc.: Groove Product Backgrounder, Corporate Whitepaper (2001).
6) Takasugi, K., Katayama, M. and Kubota, M.: Adaptive System for Service Continuity in a Mobile Environment, *7th IEEE APCC2001*, Tokyo, Japan (2001).
7) 高杉耕一，片山 穣，久保田稔，小柳恵一：ノードの移動性を実現するモビリティ技術，情報処理学会論文誌，Vol.42, No.7, pp.1828–1839 (2001).
8) Mignkhwan, A., Merabti, M. and Askwith, B.: IPMSA: Integrated Personal Mobility Services Architecture, *ICC 2002*, Vol.4, pp.2019–2023

(2002).
9) IP Mobility Support, RFC 2002 (Oct. 1996).
10) Ochi and Yamazaki: Session Mobility Protocol for Seamless Service, 信学総大, B-15-5 (2002).
11) Ramjee, R., Rorta, T.F.L., Salgarelli, L., Thuel, S. and Varadhan, K.: IP-Based Access Network Infrastracture for Next-Generation Wireless Data Network, *IEEE Pers. Commun.*, Vol.7, No.4 (2000).
12) Campbell, A., Gomez, J., Kim, S., Valko, A. and Wan, C.: Design: Implementation and Evaluation of Cellular IP, *IEEE Pers. Commun.*, Vol.7, No.4 (2000).
13) Grilo, A., Estrela, P. and Nunes, M.: Terminal Independent Mobility for IP (TIMIP), *IEEE Communications Magazine*, Vol.39, pp.34–41 (2001).
14) Snoeren, A.C. and Balakrishnan, H.: An End-to-End Approach to Host Mobility, *Proc. 6th ACM MOBICOM*, Boston, MA (2000).
15) Stoica, I., Moriis, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Abek, F.D. and Chord, H.B.: A Scalable Peer-to-peer Lookup Protocol for Internet Applications, *IEEE/ACM Trans. Networking*, Vol.11, No.1, pp.17–32 (2003).
16) 高杉耕一, 田中 聡, 中村元紀, 久保田稔：シームレスサービスにおけるユーザ利用端末の変更方法, 情報処理学会全国大会, 5K-02 (2002).
17) 高杉耕一, 中村元紀, 久保田稔：動的ネットワークにおける最適経路およびサーバサービスの選択とシームレスな切替え方法, 情報処理学会研究報告, MBL-24-29, pp.207–213 (2003).
18) 高杉耕一, 中村元紀：P2P 技術を利用したシームレスサービスプラットフォームの実現, FIT2003, M-099 (2003).
19) 高杉耕一, 中村元紀, 久保田稔, 小柳恵一：ユビキタスサービス環境に適応するサービス連続技術, 信学会学会誌, 解説 (A), Vol.86, No.2, pp.955–958 (2003).
20) Takasugi, K., Nakamura, M., Kubota, M. and Koyanagi, K.: Seamless Service Platform for a Ubiquitous and Dynamic Network Environment, *Proc. NEW2AN*, St. Petersburg, Russia, pp.4–9 (2004).
21) Diffe, W. and Hellman, M.E.: New Direction in Cryptography, *IEEE Trans. Inf. Theory*, IT-22, No.6, pp.664–654 (1976).
22) Canetti, R., Halevi, S. and Herzbeg, A.: Maintaining Authenticated Communication is the Presence of Break-ins, *Journal of Cryptology*, Vol.13, pp.61–105 (2000).
23) Diffe, W. and Hellman, M.E.: New direction in cryptography, *IEEE Trans. Inf. Theory*, IT-22, No.6, pp.664–654 (1976).
24) Canetti, R., Halevi, S. and Herzbeg, A.: Maintaining Authenticated Communication is the Presence of Break-ins, *Journal of Cryptology*, Vol.13, pp.61–105 (2000).
25) 高杉耕一, 中村元紀：動的ネットワークにおけるシームレスなサーバサービス端末の変更, FIT2002, M-89 (2002).
26) 小林正史, 高杉耕一, 梅村恭司：シームレスネットワークにおけるマルチキャスト通信の実現, FIT2002, M-55 (2002).

(平成 15 年 11 月 19 日受付)
(平成 16 年 12 月 1 日採録)

高杉 耕一（正会員）

平成 7 年東京工業大学工学部情報工学科卒業．平成 9 年北陸先端科学技術大学院大学情報科学研究科博士前期課程修了．同年日本電信電話（株）入社．平成 16 年早稲田大学大学院情報生産システム研究科博士後期課程修了．博士（工学）．現在 NTT 未来ねっと研究所研究員．アドホックネットワーク，ソフトウェア無線，アクティブタグ，モバイル環境におけるソフトウェア技術およびユビキタスサービスアプリケーションの研究開発に従事．電子情報通信学会会員．

中村 元紀（正会員）

平成 2 年名古屋大学工学部情報工学科卒業．平成 4 年同大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話（株）入社．現在，NTT 未来ねっと研究所主任研究員．通信システムにおける分散プログラム構造，無線アドホックネットワークルーティングプロトコル，およびユビキタスサービスアプリケーションの研究開発に従事．電子情報通信学会会員．

田中 聡

昭和 60 年慶應義塾大学理工学部管理工学科卒業．昭和 62 年日本電信電話（株）入社．平成 11 年から平成 16 年 3 月まで，NTT 未来ねっと研究所所属．現在，NTT ドコモネットワーク研究所主幹研究員．電子情報通信学会，ACM 各会員．

久保田 稔（正会員）

昭和 53 年東京大学工学部計計数工学科卒業．昭和 55 年同大学院工学系研究科情報工学専門課程修士課程修了．同年日本電電公社（現 NTT）入社．昭和 62 年 8 月から昭和 63 年 8 月まで米国マサチューセッツ工科大学客員研究員．平成 16 年 4 月より千葉工業大学工学部電気電子情報工学科教授．工学博士．実時間 OS，分散処理システム，ソフトウェア開発環境の研究開発に従事．電気情報通信学会，IEEE，ACM 各会員．

小柳 恵一

昭和 50 年慶應義塾大学工学部電気学科卒業．昭和 52 年同大学院修士課程修了．平成 10 年大阪大学大学院博士課程修了．昭和 52 年日本電電公社（現 NTT）入社．工学博士．現在，早稲田大学大学院情報生産システム研究科教授．著書『C ＋＋オブジェクト指向設計』,『P2P インターネットの新世紀』等．電気情報通信学会，ACM 各会員，IEEE Senior Member．