学位論文

ユーザ主導型通信制御のための
セッション層構成に関する研究

指導教員　　青山 友紀 教授

東京大学大学院情報理工学系研究科
電子情報学専攻

48-37408 金子 晋丈

目 次

図 目 次

7 セッション層情報を用いた通信資源管理

8 セッション層を用いたサービス

表 目 次

1 序論

1.1 はじめに

本章では，1.2 でインターネットを取り巻く環境の変化を概観し，インターネットを基盤としたネットワーキングの特徴として，ネットワークリソースの多様性と遍在性をあげている．1.3 では，まずインターネットの設計指針からネットワークリソースの多様化，遍在化の流れを読み解く．つづいて多様化，遍在化の流れとインターネットの設計指針との隔たりについて述べ, インターネットにおけるサービスプラットフォームの必要性について述べる．1.4 では本論文の構成を述べる．

1.2 インターネットを取り巻く環境の変化

1990 年代前半からのインターネットを取り巻く急速な技術革新には目を見張るものがある．技術革新は，インターネットに接続されるコンピューティングデバイスやインターネットを形作る通信網に限らず，様々なコンピューティングデバイスや通信網に呼応するようにアプリケーションプログラムやコンテンツにまで及んでいる．これらの技術革新は，遍在化と多様化の二つのキーワードで特徴づけられる．

インターネットに接続されるコンピュータは大型汎用機からパーソナルコンピュータに移行したと行ってよいだろう．インターネットに接続されているコンピュータの種類は，主なものを列挙するだけでも，サーバ用途のラックマウントコンピュータ，日常業務に用いるデスクトップコンピュータやラップトップコンピュータ，外出先で用いるパームトップコンピュータやセルラー端末などのパーソナルデバイス，センサー等の小型デバイス等であり，枚挙に遑がない．

インターネットに接続されるコンピュータの多様化とともに，コンピュータにインターネットへの接続性を提供するアクセス網も変化してきた．初期のインターネットにおけるネットワークは有線ネットワークで構成され，ローカルエリアネットワークには Ethernet [1] が用いられていた．その後，無線技術の飛躍的進歩と端末の小型化，無線デバイスの通信ケーブルにとらわれない高い移動性から，アクセスネットワークとして無線ネットワークが用いられるようになってきた．現在，オフィスネットワークやホームネットワークでは IEEE 802.11 [2] に基づいた無線 LAN が爆発的に普及している．一方で，セルラー端末や PHS 端末のインターネットへの接続性は，オフィスエリアやホームエリアに限らず，至る所で可能になっており，PHS による定額制のサービスも当たり前のように利用されている．

このような，コンピューティングデバイスとネットワークの多様化に伴い，インターネットを利用した通信アプリケーションプログラムやコンテンツにも多様化，遍在化の波が押し寄せている．ユーザが利用する端末やネットワークの機能的制約から，コンテンツはサービス品質の異なる同内容のものが多数存在する．WWW で公開されるコンテンツは，デスクトップコンピュータからの利用を前提にしたコンテンツだけではなく，画面の小さく接続速度の遅いセルラー端末からの利用を想定したコンテンツも同時に作られる事も多い．マルチメディアコンテンツにおいては，コーデックやエンコーディングレートの異なる複数の同一内容のコンテンツがネットワーク上で配布されている．

いつでもどこでもユーザの状況に応じたサービスを提供するために，今後，このようなコンピューティングデバイスやインターネットへのアクセス手段，およびアプリケーションプログラム，コンテンツの多様化，遍在化はさらに一層進むであろう．

1.3 インターネットにおけるサービスプラットフォーム

コンピューティングデバイスやネットワークへのアクセス手段，アプリケーションプログラム，コンテンツが多様化，遍在化は，様々なネットワークの相互接続性を高めることを設計指針にしたインターネットの初期の目的を十分に遂げた証である．すなわち，一見するとインターネットはホスト間の接続性が確保され，アプリケーションプログラムさえ開発すれば，その接続性を生かした様々な通信サービスをユーザが自由に実現できる環境が整っていることになる．

しかし，現在のインターネットはユーザのネットワーク利用を大きく制約し自由に情報をやりとりできる環境ではなくなりつつある．実際，インターネットで利用されているアプリケーションプロトコルは HTTP [3] や SMTP [4] などに大きく偏っており，これらのアプリケーションプロトコルの動作だけを前提としたネットワークの運用は決して珍しくない．また，新しいアプリケーションサービスの開発もこれらのプロトコルを前提にしている場合が多い．そして，開発されたアプリケーションサービスがネットワークの運用に影響を与え，アプリケーションプロトコルの偏りをさらに加速させている．

このようなインターネットの現状は，インターネットの設計に端を発する必然的なものである．すなわち，インターネットが様々な通信サービスやアプリケーションプロトコルを期待しつつも，それらが実際に動作する場合に必要となる共通のサービスプラットフォームを具備していないことに起因している．インターネットの設計においてはネットワークの相互接続性を高めることを第一義とした [5] ため，相互接続を困難にする要素が極力排除された．サービスプラットフォームに対応する OSI の階層モデル [6] におけるセッション層やプレゼンテーション層といったサービス層は，階層を設けることでプロトコルの制定や互換性の検証確保に多大な労力を要しインターネットの実現が困難になるとの考えから排除された [5] のである．

このような背景のもとで，突如出現した WWW 閲覧サービスとメールサービスによって HTTP と SMTP はインターネットを席捲し，その圧倒的な存在感からあたかも設計当初から存在する共通のサービスプラットフォームであるかのように利用されはじめている．これらのアプリケーションプロトコルは拡張が続けられているものの，単一のアプリケーションサービスを基盤にしているため拡張には限界があり，日々多様化し遍在化が進むアプリケーションプログラムの共通のサービスプラットフォームにはなり得ない．

本論文では，インターネットの相互接続性を最大限生かせるように，インターネットの共通サービスプラットフォームをアーキテクチャ的観点から検討を行い，これからのインターネットに必要なセッション層の構成について論じる．

1.4 本論文の構成

本論文では，インターネットの相互接続性を最大限生かしユーザが主導的に通信を制御しインターネット上で自由自在に情報を操れるようなインターネットの共通サービスプラットフォームをアーキテクチャ的観点から考察し，これからのインターネットに必要なセッション層の構成について論じている．

具体的には，まず，ユーザ主導型通信制御の意義について示し，通信制御を IP アドレス，ポート番号，トランスポートプロトコルを用いて行うことの重要性を示している．次に，通信制御機能を持つセッション層と，セッション層の認証技術について述べている．そして，セッション層が管理する通信情報の範囲を明らかにするために，IP アドレス，ポート番号，トランスポートプロトコルによって通信を管理するシステムとして IEEE 802.11 省電力モバイル端末の TCP スループット改善手法について述べている．さらに，セッション層を遠隔から制御するシステムを示し，このシステムが保持する通信制御情報をネットワーク管理に応用する通信資源管理機構を示している．最後に，セッション層を用いたアプリケーションサービスシステムを示すことでユーザ主導型通信制御を実現するセッション層の構成に関する議論を行っている．

第 2 章では, ユーザ主導型の通信制御について論じている．ユーザ主導型の通信制御とは，アプリケーションプログラムやアプリケーションサービスの制約を受けない普遍的な通信制御である．すなわち，ユーザがアプリケーションプログラムを通して情報の入手元や出力先を直接指定するのではなく，アプリケーションプログラムとは独立にユーザが通信に必要な通信チャネルを直接用意することである．さらに, ユーザ主導型通信制御を実現することで，情報をインターネット上の任意の地点から任意の地点に移動させることが可能になり，インターネットにおけるサービスをさらに豊かにする可能性があることを示している．そして，ユーザ主導型通信をインターネットで普遍的に用いるために，サービスプラットフォームとしての構築の必要性について議論している．

第 3 章では，ユーザ主導型の通信制御を実現するインターネットにおけるサービスプラットフォームとしてのセッション層について論じている．セッション層は，通信を IP アドレス，ポート番号，トランスポートプロトコルに基づいてエンドツーエンドで制御する機能である．具体的には，セッション層はアプリケーションプログラム間の通信チャネルの確立に必要な情報をセッション層より上位の階層から得るのではなく，セッション層を制御する制御部から情報を得て通信相手のセッション層との間に通信チャネルを用意し，用意した通信チャネルの端点を上位層に接合することで該当するアプリケーションプログラム間の通信を実現する．そして，ユーザがセッション層の制御部を介して通信チャネルを直接制御することで，ユーザ主導型の通信制御を実現している．

第 4 章では, セッション層が必要とするエンドツーエンドの認証技術について論じている．ユーザが構築する通信チャネルは，ユーザの要求に応じて情報の入手元もしくは出力先だけを変更できることが期待される．特に，通信チャネルの端点をホストを超えて移動させるた

めに必要となる認証技術について示している．具体的には，一つの公開鍵に対して時系列で複数の秘密鍵を対応づけられる Key-insulated 公開鍵暗号方式を用いた認証技術について示している．通信チャネルの端点を変更する際に秘密鍵を更新することで，利用した秘密鍵の流布を防ぎ，ホストに依存しないセッション層が用いる安全な認証を実現している．

第 5 章では，セッション層が管理する通信情報の範囲を，IEEE 802.11 省電力モバイル端末の TCP スループット改善手法を例に議論している．この手法は，IP アドレス，ポート番号，トランスポート層プロトコルの内部情報，をホストを超えて移動させるシステムである．本システムは，IEEE 802.11 省電力モードで動作するモバイル端末の TCP スループットの劣化を，基地局が各 TCP フローに流れる TCP セグメントを制御することで改善し，基地局間の TCP 情報の移動を実現することで，モバイル端末のハンドオーバを実現している．

第 6 章では，セッション層の制御部の機能をインターネット上のサーバに配置することについて論じている．各ホストが保持するセッション層の制御情報を，ユーザ毎にまとめて単一のサーバ上で管理することで，ユーザが複数の通信チャネルを集中的に管理制御することが可能になる．セッション層の制御情報をネットワーク上を動かない固定ホストで集中的に管理することにより，通信相手との認証処理の軽減が可能になる．さらに，セルラーネットワークを用いてこのサーバをユーザが遠隔制御することで，ユーザの利便性と安全性を兼ね備えた通信制御を実現している．

第 7 章では，ユーザ毎に管理されたセッション層の制御情報の通信資源管理への利用について論じている．セッション層の制御情報は IP アドレス，ポート番号，トランスポートプロトコルであり，これらがインターネットにおける通信資源管理の最も基本的な情報であることに着目し，インターネットの通信資源管理に利用することを論じている．具体的には，セッション層の制御情報を管理するサーバからインターネット上で通信資源管理を行うサーバにこれらの情報を伝達し，通信資源管理を行う機構を示している．特に，ローカルネットワークにおけるファイアウォールに適用し，柔軟な通信資源管理の実現を示している．

第 8 章では，サービスプラットフォームとしてのセッション層を用いたアプリケーションサービスについて論じている．セッション層は，柔軟な通信制御と集中的な認証管理，制御情報を用いた通信資源管理が可能であり，これらを統合的に利用したサービスとして遠隔会議システムを示している．具体的には，柔軟な通信制御を利用したマルチメディアコンテンツの切り替え，集中的な認証管理を用いた遠隔会議システムの利用制限，遠隔会議で用いる通信資源の管理を実現している．

1.5 むすび

本章では，ネットワークリソースの多様化，遍在化が進むインターネットを取り巻く環境の変化を概観し，インターネットの設計からインターネットにサービスプラットフォームが存在しない原因について述べ，インターネットにおけるサービスプラットフォームの必要性を議論している．そして最後に本論文の構成を示している．

参考文献

[1] Institute of Electronic and Electrical Engineers (IEEE), "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical specifications." Standard 802.3, 2002.

[2] Institute of Electronic and Electrical Engineers (IEEE). "Wireless medium access control (MAC) and physical layer (PHY) specifications." Standard 802.11, 1999.

[3] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee. "Hypertext transfer protocol — HTTP/1.1," RFC 2616, Internet Engineering Task Force, June 1999.

[4] J. B. Postel, "Simple mail transfer protocol," RFC 821, Internet Engineering Task Force, August 1982.

[5] J. B. Postel, "Internetwork protocol approaches." IEEE Transactions on Communications, Vol.28, No.4, pp.604-611, April 1980.

[6] H. Zimmerman, "OSI reference model — the ISO model of architecture for open system intereconnection," IEEE Transactions on Communications, Vol.28, No.4, pp.425-432, April 1980.

2 ユーザ主導型通信

2.1 はじめに

ユーザ主導型通信は，情報の遍在化が進むこれからのコンピュータネットワークにおける新しいネットワーキングのスタイルである．具体的には，現在のアプリケーションプログラムを通した制御から通信の制御を完全に分離することで，ネットワーク上に遍在する情報の流れをユーザが直接制御して情報の発信，加工，受信を実現しようとするものである．

本章では，2.2 でユーザ主導型通信の意義およびユーザ主導型通信によって実現される様々な通信サービスを，2.3 でインターネットにおけるユーザ主導型通信について述べる．

2.2 ユーザ主導型通信の意義

情報は情報源に存在しているだけでは情報として意味を持たない．すなわち，情報は情報源から情報源とは異なる場所に移動してはじめて情報としての価値が発生することになる．そして，情報を移動させることが通信であり，情報を移動させる手段が通信手段に他ならない．

一方で，情報は受け取ったままの形では十分には活用しきれない．すなわち，受け取った情報は，情報を扱う者が目的に応じて整理し適切な形に加工する必要がある．この整理と加工が情報処理である．そして，情報処理の過程においては，異なった情報処理ツールを用いて加工，整理を複数回繰り返すことも多く，この情報処理ツール間の情報の受け渡しもまた通信である．

近年のコンピュータの発展は，情報処理の技術を格段に進歩させてきたとともに，コンピュータネットワークを構築することで通信手段としても用いられるようになってきた．コンピュータが情報処理ツールとしてだけではなく通信手段としても利用されることは，情報をコンピュータで一貫して扱うことが可能になったということであり，われわれが情報を自由自在に扱える環境が整ったことを意味している．

すなわち，通信を行うということは，コンピュータネットワーク上にある情報処理ツールとしての無数のアプリケーションプログラムの中から一つアプリケーションプログラムを選択して，通信相手が送信してくる情報を受け取ることである．もしくは，アプリケーションプログラムを一つ選択して，通信相手に対して情報を送信することである．情報処理を行うということは，情報を出力するアプリケーションプログラムを指定し，情報を受け取るアプリケーションプログラムを自ら指定することに他ならない．

したがって，コンピュータが普及した環境における情報の制御は，コンピュータネットワーク上のアプリケーションプログラム間をどのように繋ぐかだけで決まると言っても過言ではない．ユーザがアプリケーションプログラムを制御するのではなく，アプリケーションプログラム間の接続性を制御して情報の制御を行うこと，これが通信制御の意義である．

例えば，ユーザが留守の間に来客があった場合，玄関先の様子をユーザの手元のディスプレイに表示するといったことも，玄関先に設置したカメラとユーザの手元にあるディスプレイをユーザが接続するだけで簡単に実現できる．すなわち，ユーザのコンテキストの変化に応じて，ユーザが通信チャネルを新しく作ったり切り替えたり，終了させるだけで，アプリケーションプログラムを改変したり制御したりすることなく，所望のアプリケーションサービスを実現することが可能になる．

本論文が目的とするユーザ主導型通信制御は，上述の通信制御を単にユーザが行うのではなく，ユーザが制御できる通信制御の範囲を厳格に設けている．これは，通信は一般に異なるユーザの間で行われるものであるという前提に立脚している．

通信が異なるユーザの間で行われるということは，以下の二つの制約を持つことになる．

ひとつは，異なるユーザ間で行われる通信は，両ユーザの同意があってはじめて実現され，

同意が崩れたら通信は中断され終了されなければならないということである．

もうひとつは，ユーザが制御できるのは通信相手に送信前の情報の処理と送信するアプリケーションプログラムの選択，もしくは通信相手から受信するアプリケーションプログラムの選択と受信後の情報の処理であり，送信後の処理や受信前の処理に関して通信相手にいかなる制約も与えないということである．

上記の制約をもつユーザ主導型通信制御を行うことで，コンピュータを基盤にした情報環境を最大限に活用することが可能になる．

例えば，コンテンツデリバリー会社がユーザ側アプリケーションプログラムのネットワーク位置やコンテンツサーバの混雑具合に応じて，より適切なコンテンツ配信サーバを切り替えるということも，ユーザ側に負担をかけることなく自在に行うことができる．これは，ユーザにコンテンツを配信するコンテンツ配信サーバを選択するのはコンテンツデリバリー会社であり，ユーザではないということが明確化されているからである．

別の例として，映画視聴サービスプロバイダと 24 時間視聴サービスを契約したユーザがいたとしよう．映画視聴サービスプロバイダはユーザと視聴時間が合計 24 時間になると視聴が中止されるという合意をしているものとする．このとき，映画視聴サービスプロバイダはユーザの視聴時間が 24 時間を超えた時点でユーザとの間の通信を切断し，以後一切の通信を拒否することが可能になる．これは，ユーザ間の契約そのものを通信の生存に直接対応させており，多くのビジネスシーンで活用されることになろう．

2.3 ユーザ主導型通信とインターネット

ユーザがアプリケーションプログラムを指定しアプリケーションプログラム間の通信チャネルをユーザが直接構築することで，アプリケーションプログラムに通信相手を伝えることなくアプリケーションプログラム間の通信を実現することを，ユーザ主導型通信と呼ぶ．

単一のコンピュータ内に限定するとユーザ主導型通信はプロセス間通信として古くから実現されている [1]．プロセス間通信において，ユーザはシェルを通してアプリケーションプログラムを指定し，シェルがプログラム間の通信チャネルを構築するため，ユーザ主導型通信であるといえる．例えば, パイプ” |” やリダイレクション”>, >>” があればアプリケーションプログラムに与える情報を変えなくともデータの出力先を変更することが可能であり，パイプを多段に組めば複数のアプリケーションプログラムの間で順にデータを受け渡しながら情報を加工することができる．

しかし，プロセス間通信はアプリケーションプログラムが同一の親プロセス（シェル）を持たなくてはいけないという制約 [1] があり，DARPA Internet 等の通信ネットワークを介して接続される分散データベース等への適用ができなかった．そのため，アプリケーションプログラムがどのコンピュータ上で動作していても透過的に通信ができるように，アプリケーションプログラムが直接通信相手の” 名前” を指定するプロセス間通信が設計された．設計の背景には，通信のモジュール化による利便性よりも通信の効率を重んじるという設計指針が存在している [2]．

現在の通信は，上述の通信モデルをそのまま用いているため，ユーザ主導型通信を実現できていない．すなわち，インターネットではアプリケーションプログラムを介した通信制御しかできないため，単一のコンピュータ内ではパイプなどによってアプリケーションプログラムに依存せず簡単に実現できることが，複雑なアプリケーションプログラムを開発した上で実装間のプロトコルの整合性を確保しなければ実現することができない．

したがって，現在のインターネットにおける通信では，通信制御のユーザ主導性と，通信制御がもたらす情報制御によって得られる利便性を失っていることになる．

一方で，インターネットはユーザ主導型通信制御に適した通信ネットワークである．ユーザ主導型通信は，前節で述べたように異なるユーザ間の通信はユーザ間の同意に基づいて行われるものであるため，それぞれのユーザが通信の制御権を保有する必要がある．そして，インターネットはエンドツーエンドの思想に基づいて構成されている [3] ため，ユーザは利用するエンドノードに直接指示するだけでユーザの独立した制御権の確保が簡単に実現できる．

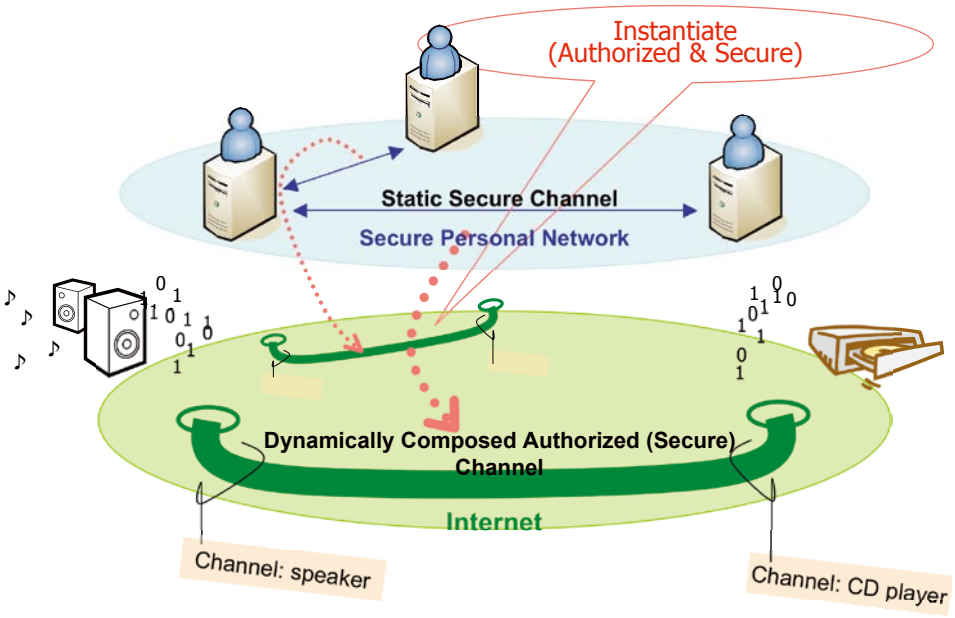


図 2.1 ユーザ主導型通信制御

2.4 むすび

本章では，ユーザ主導型通信の意義について述べた，また，インターネットにおけるユーザ主導型通信の欠如を，コンピュータネットワークの発展の経緯の観点から述べた，さらに，ユーザ主導型通信制御とインターネットの親和性について述べた．

参考文献

[1] W. R. Stevens, "UNIX network programming, volume 2, second edition: interprocess communications," Prentice Hall, 1999, ISBN 0130810819.

[2] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarteman, "The design and implementation of the 4.4BSD operating system", Addison-Wesley, 1996, ISBN 0201549794.

[3] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," ACM Transactions on Computer Systems, pp.277-288, 1984.

3 セッション層の導入

3.1 はじめに

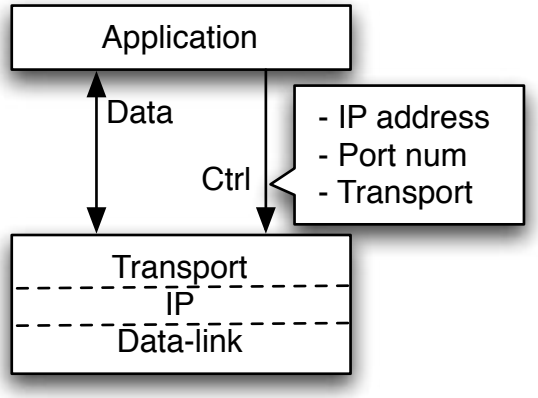
広く普及したインターネットでは，期待される様々な通信サービスやアプリケーションプロトコルが実際に動作する場合に必要となる共通のサービスプラットフォームの構築が不可欠となる．

筆者の目指す共通サービスプラットフォームとは，伝達する情報の種類によらない普遍的な情報流通のフレームワークと通信相手との認証および安全な通信を保証するセキュリティフレームワークを兼ね備えるものである．本論文では，インターネットの相互接続性を最大限生かした共通サービスプラットフォームの構築についてアーキテクチャ的観点から検討を行い，これからのインターネットに必要なユーザ主導型通信制御を実現するセッション層について論じる．

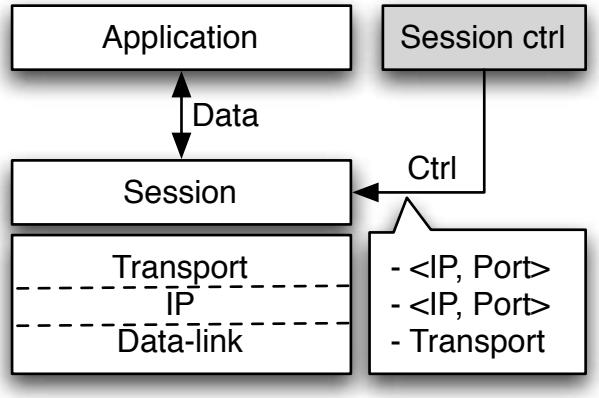
本論文で述べるセッション層は，IP アドレス，ポート番号，トランスポート層プロトコルで識別される通信チャネルを制御する．セッション層における通信制御とは，アプリケーションプログラムが情報の入手元や出力先を直接指定するのではなく，セッション層がアプリケーションプログラムとは独立に通信に必要な通信チャネルを用意することである．具体的には，セッション層はアプリケーションプログラム間の通信チャネルの確立に必要な情報をセッション層より上位の階層から得るのではなく，セッション層を制御する制御部から情報を得て通信相手のセッション層との間に通信チャネルを用意し，用意した通信チャネルの端点を上位層に接合することで該当するアプリケーションプログラム間の通信を実現する (図 3.1.1)．

ユーザがセッション層の制御部を介して通信チャネルを直接制御することで，ユーザ主導型の通信制御を実現し，インターネットの共通サービスプラットフォームが構築できる．すなわち，ユーザはインターネット上に遍在するアプリケーションプログラム間をアプリケーションプロトコルやアプリケーションプログラムの制約を受けずに接続し，一端にあるアプリケーションプログラムが出力する情報を他端のアプリケーションプログラムに受け渡すことが可能になる．これは，情報の入手元や出力先をアプリケーションプログラムを通して指定するアプリケーションプロトコルへの依存性が強い現在の方法と異なり，普遍性の高い通信制御を実現しておりインターネットにおける情報流通を促進する．同時に，ユーザが通信チャネルの構築や解放を直接判断することになるため，ユーザ主導型のセキュリティフレームワークを兼ね備えることができる．

以下では，3.2 でユーザ主導型通信制御を実現にあたっての設計指針について述べる．次に，3.3 でユーザ主導型通信制御を実現するセッション層の詳細について示し，3.4 でセッション層の実装を，3.5 で実装に基づいた評価を行う．3.6 で関連研究をまとめる．



(a) 現在の通信モデル



(b) セッション層を導入したモデル

図 3.1.1 セッション層の導入

3.2 ユーザ主導型通信制御の設計指針

インターネットにおけるユーザ主導型通信の実現にあたり，三つの設計指針–インターネットとの親和性，ユーザの主導性，アプリケーションサービスの多様性–を挙げる．

3.2.1 インターネットとの親和性

ユーザ主導型通信をインターネットで実現する上で最も重要な点は，インターネットとの親和性を損なわないことである．

・エンドツーエンド

ユーザ主導型通信はサービスプラットフォームであり，必ずしもインターネット上のすべてのホストが必要とする機能ではないため，ネットワーク層の機能としてではなくエンドホストの機能として実現しなければならない [1].

仮に，ネットワーク層に機能を追加する方法やオーバレイネットワークを構築しオーバレイネットワークのネットワーク層の機能として実現する方法を採用した場合，名前やアドレス解決のオーバヘッドが発生しルーティング効率が低下する [2, 3, 4]．結果としてサービスプラットフォームとしての効率性を損なうことになる．また，ネットワーク層への機能追加は，ネットワーク全体への展開が必要になるため普及が困難である．

・既存のアプリケーションプログラムとの共存

インターネット上で動作するアプリケーションサービスは，他のアプリケーションサービスの動作を制約しないことが望まれる．ユーザ主導型通信を実現するサービスプラットフォームも，インターネットのひとつのアプリケーションサービスであり，これを動作させることによって既存のアプリケーションサービスの動作を阻害することがあってはならない．

3.2.2 ユーザの主導性

ユーザの主導性を確保するためには，アプリケーションプログラムに依存しない通信認証とユーザによる通信チャネルの伝送特性の決定が必要である．

・通信認証

通信は二人のユーザが対等な立場で実現するものである．

したがって，通信におけるユーザの主導性を確保するには，アプリケーションプログラムの利用に関する認証と，アプリケーションプログラム間の通信チャネル利用に関する認証が完全に分離されなければならない．すなわち，アプリケーションプログラムは常に単一のユーザの指示に従って動作し，アプリケーションプログラムを繋ぐ通信チャネルは，アプリケーションプログラムとは独立に両端のユーザの同意に基づいて利用されなければならない．

現在のインターネットでは，上記の二つの認証が完全に分離されていない．例えば HTTP [5] のベーシック認証では，利用するウェブブラウザがユーザの入力した認証情報を悪用しないことを前提にして，ユーザはウェブブラウザにウェブサーバがコンテンツにアクセスするためのユーザ名とパスワードを入力している．しかし，アプリケーションプログラムが遍在する環境では，すべてのアプリケーションプログラムの信頼性を保証することは困難である．

また，アプリケーションプログラムとは別の安全性が確認されたプログラムから通信相手のアプリケーションプログラムとの間で認証を行う場合は，あらかじめ通信相手のユーザが利用するアプリケーションを知っていることが前提となる．この前提はアプリケーションサービスの制約に繋がる．

したがって，まず，ユーザはそれぞれのユーザの権限でアプリケーションプログラムを動作させ，通信相手となるユーザとの間でそれぞれが安全性を確認したプログラムを用いて認証を行う．認証後，ユーザが連携してアプリケーションプログラム間に通信チャネルを構築し，認証に基づいてユーザがそれぞれのアプリケーションプログラムにアクセスレベルを設定する必要がある．

・通信チャネルの伝送特性の決定

通信チャネルの伝送特性はユーザの意図に基づいて決定されなければならない．伝送特性とは，具体的には通信チャネルに用いるトランスポート層プロトコルと通信チャネルの QoS および暗号化の指示である．

トランスポート層プロトコルはユーザのネットワーク環境によって性能が大きく左右されるため，ユーザの判断に基づいてトランスポート層プロトコルを切り替えることで通信の性能を向上させることが可能となる．例えば，無線環境では TCP のスループットが低下することはよく知られている [6]．そこで，アプリケーションプログラムがトランスポート層プロトコルに TCP を使うように指定していても，通信するホストが無線環境に適したトランスポート層プロトコルをサポート [7] しているのであれば，ユーザがこのプロトコルを指定することで無線環境であっても通信性能が向上することになる．

同様に，QoS が保証された通信を実現するためには，通信チャネルの QoS をユーザが決定できなければならない．現在のインターネットでは QoS を考慮したルーティング [8, 9] が実現されていないため，これは，QoS がサポートされたネットワークで用いることになる．

また，秘匿性の高い通信を実現するために，通信チャネルの暗号化もアプリケーションプログラムと無関係に実現できなければならない．

3.2.3 アプリケーションサービスの多様性

ユーザ主導型通信は，アプリケーションサービスの種類によらず動作することが求められるため，アプリケーションサービスの多様性を損なってはならない．

・通信チャネルのトランスポート層プロトコル

ユーザ主導型通信に基づいて多様なアプリケーションサービスを利用できるようにするため, 通信チャネルは特定のトランスポート層プロトコルに制約されてはならない．すなわち，アプリケーションプログラムが指定する任意のトランスポート層プロトコルで通信チャネルを構築できなければならない．

これは，トランスポート層プロトコルによって伝送特性が大きく異なり，トランスポート層プロトコルを制約することがアプリケーションサービスの制約に繋がるからである．例えば，TCP はインターネット上の任意の２点間の信頼性のあるデータ伝送を行うが，再送を行うためリアルタイム性の高い音声・画像データの送受信には向かない．ユーザ主導型通信がトランスポート層プロトコルとして TCP だけをサポートした場合，リアルタイム性の高いアプリケーションサービスで通信のユーザ主導性を実現することができなくなる．

・アプリケーションプログラムの開発

多様なアプリケーションサービスの普及と発展を図るには，簡単にユーザ主導型通信のサービスフレームワークを利用できる必要がある．また，サービスフレームワークの拡張や仕様変更に伴うアプリケーションプログラムの再開発や再コンパイルを要さないようにする必要がある．

3.3 セッション層

3.3.1 セッション層の導入

ユーザ主導型通信を実現するためにセッション層を導入する. セッション層は, アプリケーションプログラムをネットワークから完全に分離することでユーザの主導性を確保する．また，セッション層は，その名称からも明らかなようにトランスポート層よりも上位の階層で動作するサービスフレームワークであり, アプリケーションプログラムとは独立に動作する. すなわち，セッション層は，通信チャネルをインターネット上の任意のコンピュータ間で動的に構築できるようにするとともに，構築した通信チャネルをアプリケーションプログラムに動的に接合する機能を持つ.

セッション層としてユーザ主導型通信を実現する理由は，以下の二つである．ひとつは，通信チャネルのユーザ主導性やアプリケーションサービスの多様性を実現するためには，トランスポート層プロトコルを使い分ける必要があり，単一のトランスポート層プロトコルとしての実現は困難であるからである．もうひとつは，アプリケーションプログラムへの依存性を排除するためには，アプリケーションプログラムとは独立して動作する必要があるからである.

セッション層の導入により，アプリケーションプログラムはネットワークに関する情報を直接扱わなくなる．セッション層ミドルウエアの導入によって実現される通信モデルは，これまでの通信のモデルとは以下の三つの点で異なる.

・ネットワークからのアクセス

アプリケーションプログラムはセッション層からの接続要求のみを受け付け，ネットワークからの接続要求を受け付けないため，端末外部からはアプリケーションプログラムの動作が確認できない. したがって, セッション層は動作アプリケーションプログラムを把握しユーザに適切に伝える機構が必要になる．一方，アプリケーションプログラムはネットワークに接続した状態にないので，ポートスキャンや SYN 攻撃などのネットワークからの攻撃を受けることはない.

・通信のモビリティ

アプリケーションプログラムはセッション層と通信するだけであり，IP アドレスやポート番号等の情報を扱わないためモビリティへの対応を必要としない．セッション層がモビリティサポートの機能を保持すれば，端末の IP アドレスが変化するターミナルモビリティや，通信相手のアプリケーションプログラムが切り替わるサービスモビリティが実現できる.

・通信の効率

アプリケーションプログラムはセッション層を介して通信を行うため通信の効率が劣化す

る [10]．これまでは，アプリケーションプログラムが直接 OS と送受信データを受け渡していたが，セッション層の導入により，アプリケーションプログラムはセッション層との間でデータをやりとりしセッション層が OS との間でデータをやりとりしなければならないからである．セッション層の機能を OS 内部に組み込むと通信の効率は改善するが，広く普及した OS へのセッション層の導入は容易ではない．

3.3.2 セッション層の概要

ユーザ主導型通信を実現するセッション層は，セッション層ミドルウエアとセッション層制御部によって構成され，アプリケーションプログラムはセッション層 API を用いてセッション層の機能を利用する（図 3.3.1）．

セッション層ミドルウエアはセッション層制御部からの指示を受けて動作する．具体的なセッション層ミドルウエアの動作は，通信相手のセッション層ミドルウエアとの間の通信チャネルの構築破棄と，同じ端末上で動作するアプリケーションプログラムとの間の通信チャネルの構築破棄，およびこれら二つの通信チャネルに流れるデータをプロキシ処理である．

一方，セッション層制御部はユーザの指示に基づいて動作する．具体的なセッション層制御部の動作は，通信相手のセッション層制御部と認証処理を行い通信チャネルの構築に必要な情報を交換し，セッション層ミドルウエアに，通信チャネルの構築破棄の要請とその通信チャネルを使うアプリケーションプログラムを指示する．

以下では，セッション層について，アプリケーションプログラムが利用するセッション層 API，セッション層ミドルウエア，およびセッション層制御部の三つに分けて説明する．

3.3.3 セッション層 API

アプリケーションプログラムはセッション層ミドルウエアとの通信にセッション層通信インターフェース（slsocket）を利用する．

slsocket は，アプリケーションプログラムがセッション層の機能を簡単に利用できるようにするための通信インタフェースであり，アプリケーションプログラムはセッション層 API（slsocket API）によってセッション層の機能を利用することができる．

slsocket はコネクション型通信を行い，コネクションは常にセッション層ミドルウエアからアプリケーションプログラムに対して構築される．すなわち，アプリケーションプログラムは，セッション層ミドルウエアからの通信を待ち受ける状態で動作し，セッション層ミドルウエアとの接続が確立すると，データの送受信を開始する．slsocket をコネクション型通信にすることでアプリケーションプログラムは通信相手の存在を把握することが可能になり，通信相手がいないのにデータを送信し続けることやデータの受信を試みることがなくなる．

セッション層ミドルウエアがアプリケーションプログラムに接続を行うためには，アプリケーションプログラムの待ち受けインタフェースの " 名前 " 情報が必要になる [10]．" 名前 "

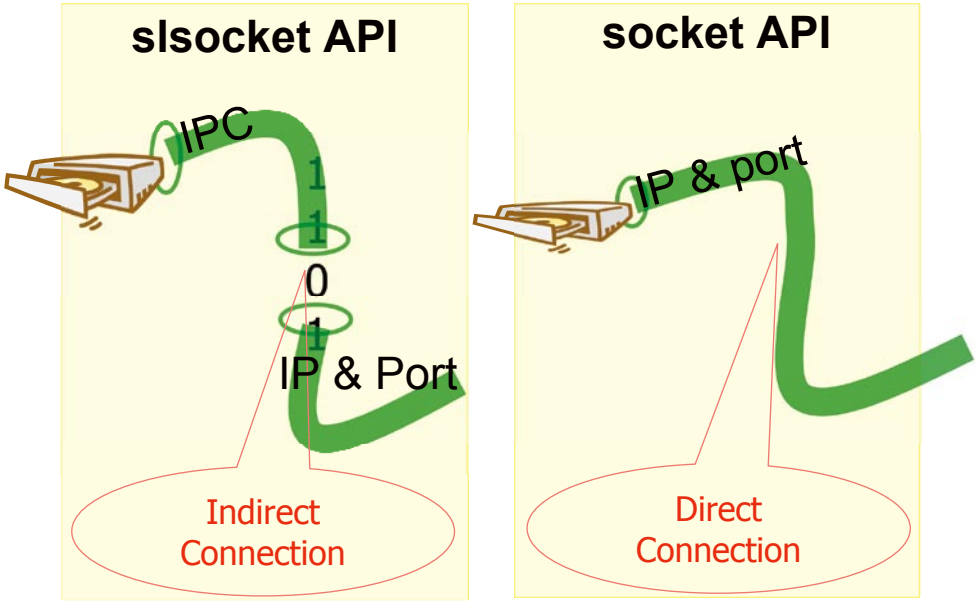


図 3.3.1 セッション層接続

情報とは具体的には，AF_INET ドメインや AF_INET6 ドメインで通信する場合には待ち受けポート番号であり，AF_UNIX ドメインで通信する場合には待ち受けファイルパスである．これらの " 名前 " 情報は現在のポート番号のようにアプリケーションサービスに固有の情報である必要はない．そのため，" 名前 " 情報は，アプリケーションプログラムが slsocket API を通して slsocket を作成したときに API 内部で自動的に生成され，さらにセッション層ミドルウエアに slsocket が提供するサービス内容とともに通知される．セッション層ミドルウエアは，アプリケーションプログラムから通知されたサービス内容と " 名前 " 情報を管理し，セッション層制御部からの指示に応じて，現在利用可能なサービス内容の提示，接続要求に対応する．

アプリケーションプログラムとセッション層ミドルウエア間の通信では，slsocket API がアプリケーションプログラムの送受信データサイズを示すヘッダを付与する．これは，アプリケーションプログラムがインターネットにおけるトランスポート層プロトコルとして UDP 等のパケット毎のデータ境界を必要とするデータグラム通信を希望する事を想定し，そのデータ境界をセッション層ミドルウエアとアプリケーションプログラム間で共有するために行う．なお，このヘッダは slsocket 内部で用い，インターネットに送出される際には除去される．

3.3.4 セッション層ミドルウエア

セッション層ミドルウエアは，セッション層制御部からの指示に従って対向セッション層ミドルウエアとの接続に用いる通信インタフェースを構築する．この通信インタフェースは，アプリケーションサービスに固有の情報を一切含まず，ポート番号は空きポート番号から動的に決定される．このため，セッション層ミドルウエアはセッション層制御部から利用可能 IP アドレスや利用可能ポート番号について問い合わせを受け，セッション層制御部からの指示に従って，通信インタフェースを構築する．

セッション層ミドルウエアが構築する通信チャネルは，セッション層制御部から指定された IP アドレスとポート番号からのデータしか受け付けない．通信チャネルのトランスポート層プロトコルとして TCP を用いる場合は，コネクションを accept するときに，セッション層制御部に指定された IP アドレス，ポート番号でなければ TCP コネクションを切断し，UDP を用いる場合は，パケットの送信元 IP アドレスとポート番号をパケット毎にチェックし，意図しないパケットは破棄する．

対向セッション層ミドルウエアとの通信インタフェースが構築されアプリケーションプログラムとの slsocket が用意されると，セッション層ミドルウエアは，それぞれのインタフェースから受け取ったデータのプロキシ処理を開始する．セッション層ミドルウエアはプロキシ処理のための特別のバッファ [11] を持たず，セッション層ミドルウエアに到着したデータを到着した順にもう一方のインターフェースにデータを書き込むことでプロキシ処理を実現する．

バッファを設けずにプロキシ処理を行うのは，アプリケーションプログラム間のデータの

やりとりの即時性が失われる可能性があるからである．これまでの通信ではアプリケーションプログラムがデータを通信インタフェースに書き込むと，書き込まれたデータが滞留する可能性のある箇所は，自端末の OS 内部と通信相手端末の OS 内部にしかなかった．しかし，セッション層を用いるとアプリケーションプログラム間の通信は合計三つのコネクションに分割され，アプリケーションプログラムが slsocket に書き込んだデータの滞留する箇所は，slsocket 通信とネットワークを介した通信のそれぞれのために自端末の OS 内部の滞留が発生し，通信相手端末でも同様に二箇所で滞留が発生する．これにバッファを加えると合計六箇所で滞留することになる．滞留箇所が増えると，例えば受信側のアプリケーションプログラムのデータ読み取り速度が遅くバッファ欠乏状態になっても，送信側のアプリケーションプログラムがそれを検知するまでに長い時間を要することになる．

3.3.5 セッション層制御部

セッション層制御部は，通信相手のセッション層制御部とユーザ認証を行い，セッション層ミドルウエアが構築する通信チャネルの構築に必要な情報を交換し，セッション層ミドルウエアに指示する．

セッション層制御部が行う認証はユーザ間の認証であり，アプリケーションサービスによって求められる認証方法が異なるため，特定の認証方法に制約しない．例えば，あらかじめ安全に公開鍵の交換ができていることを前提とした直接認証はユーザの個人的な人間関係の認証に適している．また，PKI 等を用いた第三者認証を用いる方法は，一方のユーザがアプリケーションサービスプロバイダであるときの認証に適している．

セッション層制御部は対向のセッション層制御部との間で制御メッセージをやりとりする．通信開始時の制御内容は，ユーザが利用するアプリケーションサービスとセッション層ミドルウエアが利用する IP アドレスのネゴシエーション，セッション層ミドルウエアが利用するポート番号の通知，TCP 等のコネクション型通信を用いる際の待ち受け・接続側の指示，通信チャネルを暗号化する際の鍵の通知，通信チャネル確立完了の通知である．通信終了時の制御内容は，通信終了の要請である．それぞれの制御は対向のセッション層制御部からの制御応答によって確認される．

セッション層制御部間の通信は暗号化することが望ましい．暗号化することで，どのような通信チャネルを構築しようとしているのかを外部に漏らさないようにすることが可能になる．これにより，通信チャネル構築前に通信チャネルの構築に必要な情報が漏れて，意図しないユーザが通信チャネルの構築を試みることを防ぐことができ，通信チャネルのユーザ主導性を確固たるものにすることができる．セッション層ミドルウエアが構築する通信チャネルのポート番号はオンデマンドに決定され，制御メッセージが暗号化されるため，意図しない通信チャネルが構築される可能性は非常に低い．なお，構築した通信チャネルが通信途中でハイジャックされる可能性は否定できないが，これは現在のインターネットにおける通信ハイジャックの可能性と同じである．これに対しては，通信チャネルを暗号化することで改善される．

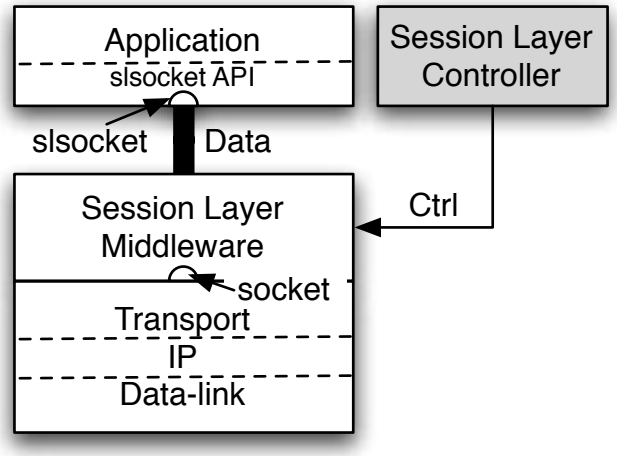


図 3.3.2: セッション層構成図

3.4 実装

セッション層を NetBSD2.1 と Windows XP 上に C 言語で実装した．

3.4.1 セッション層 API

図 3.4.1 に，slsocket API を用いたセッション層のサンプルコードを示す．socket API を用いた場合は TCP のクライアント側とサーバ側のプログラム方法の相違によって別々のプログラムを用意しなければならないが，slsocket API を用いた場合は通信インタフェースの作成方法はトランスポート層プロトコルに依らず図に示す方法のみである．

slsocket_initialization により待ち受け slsocket の " 名前 " 情報と slsocket が提供するサービス名をセッション層ミドルウエアに登録する．sl_listen はセッション層ミドルウエアからの接続を待ち受ける関数で，sl_accept によりセッション層ミドルウエアからの接続を確立する．sl_send，sl_recv は，slsocket へのデータの読み書きを行う関数である．sl_close は，sl_accept した slsocket を終了するための関数であり，slsocket_close は，セッション層ミドルウエアから当該 slsocket の登録を解除する．いずれの関数も，アプリケーションプログラマに配慮し既存の socket 関数に似せた API として実装した．本実装においてアプリケーションプログラマが扱う slsocket は，ファイルディスクリプタになっており，既存の通信インタフェースと同様に select を用いた受信確認やタイマ設定等が可能である．

3.4.2 セッション層ミドルウエア

セッション層ミドルウエアは，大きくアプリケーションサービス登録部と通信制御受付部に分かれている．

アプリケーションサービス登録部は，セッション層を用いるアプリケーションプログラムを管理するモジュールである．slsocket_initialization がアプリケーションプログラムによって呼ばれると，slsocket API が内部処理としてセッション層ミドルウエアにアプリケーションサービス登録メッセージを送信する．アプリケーションサービス登録部は，各アプリケーションプログラムのプロセス番号と slsocket のファイルディスクリプタに基づいて slsocket 情報を管理するテーブルを構築する．テーブルには，slsocket_initialization 時にアプリケーションプログラムが設定したアプリケーションサービス名，slsocket API が自動生成するアプリケーションサービスの待ち受け socket の名前情報が含まれる．アプリケーションプログラム登録部は，定期的にこのテーブルに記載されたアプリケーションプログラムのプロセス番号に基づいて，そのプロセスが継続して動作しているかを確認し，すでに修了している場合は，テーブルから消去する．また，アプリケーションプログラムが slsocket_close を呼ぶと，slsocket API は再びセッション層ミドルウエアに対して slsocket 修了メッセージを送信し，該当 slsocket 情報はテーブルから削除される．登録されているアプリケーションサービスの情報は，アプリケーションサービス登録部に一覧出力要求を送信することで得られる．

```
{
        int slsock, accept_slsock;
        struct slsocket_description {
                char description[32]; //service description
                u_short option;
                int transport_type;
        } sld;

        /* make a new slsocket */
        memset(&sld, 0, sizeof(struct slsocket_description));
        snprintf(sld.description, sizeof(sld.description),
                        "slsocket sample");
        sld.transport_type = SL_TRANSPORTTYPE_DGRAM;
        // sld.transport_type = SL_TRANSPORTTYPE_STREAM;
        if (slsocket_initialization(&slsock, &sld) < 0) {
                fprintf(stderr,
                        "Err: Session Layer Middleware is not loaded\n");
                return -1;
        }
        /* start listening on the slsocket */
        if (sl_listen(slsock, 5) < 0) {
                fprintf(stderr, "Err: sl_listen\n");
                goto ERR2;
        }
        /* accept a new slsocket */
        memset(&sld, 0, sizeof(struct slsocket_description));
        accept_slsock = sl_accept(slsock, &sld);
        if (accept_slsock < 0) {
                fprintf(stderr, "Err: sl_accept\n");
                goto ERR2;
        }
        /* write data on slsocket */
        write_size = sprintf(write_buf, "Hello, session world.\n");
        if (sl_send(accept_slsock, write_buf, write_size) != write_size)
                fprintf(stderr, "Err: sl_send\n");
        /* read data from slsocket */
        memset(read_buf, 0, sizeof(read_buf));
        read_size = sl_recv(accept_slsock, read_buf, sizeof(read_buf));
        if(read_size == 0) {
                fprintf(stderr, "Err: peer closes slsocket\n");
                goto ERR1;
        } else if(read_size < 0) {
                fprintf(stderr, "Err: sl_recv\n");
                goto ERR1;
        } else {
                read_buf[read_size] = '\0';
                fprintf(stdout, "%s\n", read_buf);
        }
ERR1:
        /* close slsocket accepted */
        sl_close(accept_slsock);
ERR2:
        /* close listening slsocket */
        /* (correspondent to slsocket_initialization) */
        slsocket_close(slsock);
}
```

図 3.4.1: slsocket API を用いたサンプルアプリケーションプログラム

通信制御受付部は，セッション層制御部からの指示を受けて，通信相手のセッション層ミドルウエアとの間の socket の作成，アプリケーションプログラムとの接続を行い，これら二つのインタフェースに流れるデータをプロキシ処理する．実際の処理は，一つの通信チャネルに対して一つのスレッドが新たに作成される，すなわち，このスレッドは，セッション層制御部からの制御用インタフェース，アプリケーションプログラムとの通信インタフェース, 通信相手のセッション層ミドルウエアとの通信インタフェースを保持し, socket の作成, bind 処理, listen, accept/connect 処理, プロキシ処理, 終了処理からなる一連の処理を担っている．

3.4.3 セッション層制御部

セッション層制御部は，他の通信端末のセッション層制御部とユーザ認証および通信チャネル確立に必要な情報の交換を行う．セッション層制御部の処理手順を以下に示す．

まず，セッション層制御部間で認証を行い，サービス開始希望を通信相手側セッション制御部に送信する．ユーザ B 側が通信の許可を発行すると，セッション層ミドル間の通信チャネルの設定の制御メッセージのやりとりが行われる．ユーザ B の通信許可には利用可能な IP アドレスの候補が記載されており，A 側は通信に利用する IP アドレスペアを決定し，A 側のセッション層ミドルウエアと B に対して bind 要求を送信する．bind 完了メッセージが双方から返ってくると，片側に listen 要求を送信する．listen 要求時には IP アドレス情報に加えて bind 完了メッセージで得られたポート番号を合わせて送信し，accept する通信チャネルを指定する．listen が完了すると connect 要求をもう一方に送信する．connect 要求時には，bind 完了メッセージに記載された通信相手のポート番号を記述する．listen 要求と connect 要求は，セッション層がトランスポート層プロトコルに依存しないようにするため，通信チャネルのトランスポート層プロトコルが TCP であるか UDP であるかに関わらずに行われる制御である．トランスポート層プロトコルが TCP の場合は，connect か accept が完了した段階で，通信チャネル構築完了メッセージが通信制御サーバに届く．UDP の場合は，通信チャネル構築完了メッセージは，listen 要求と connect 要求の完了とともに届く．

3.5 評価

セッション層の性能評価として，通信スループット性能と CPU 使用率を測定した．以下の測定では Windows XP を用いている．

3.5.1 スループット性能

slsocket API 適用時，非適用時において，トランスポートプロトコルに TCP，UDP を利用した場合のスループットを測定した．測定時間がおよそ 1 分になるように，測定サイズに応じて 5 万パケットから百万パケットの間で変動させ，それぞれ 10 回ずつ測定を行い，その平均を算出した．TCP, UDP における slsocket API 適用時, 非適用時におけるスループット性能を，それぞれ図 3.5.1，図 3.5.2 に示す．

バルクデータ転送サイズが約 1,000 バイトまでは，slsocket API 適用時，非適用時のスループットに差が出た．slsocket API 適用時のスループットの低下は，コンテキストスイッチ，メモリコピーのオーバヘッドに起因していると考えることができる．また，Windows XP の初期状態では，UDP パケットが 1,024 バイトを超えるとコンテキストスイッチの総数と CPU 使用率が増え，供給できる最大クライアント数が減少する．この問題を解決するために，測定ではレジストリ値 FastSendDatagramThreshold を 1,024 から 65,536 に変更した．この変更に伴い, 1,024 バイトまでのスループットが変更前と比較して低下したため，UDP のスループットが TCP のスループットを下回っている．

3.5.2 CPU 使用率

同様に，CRN Monitor [12] を用いて実通信端末の CPU 使用率を測定した．この測定では，バルクデータ転送サイズが 64 バイト，1,024 バイト，8,192 バイトの場合を比較した．受信側において測定した結果を図 3.4.3，図 3.5.4 に示す．また，送信側の CPU 使用率は slsocket API 適用時，非適用時とも常時 100% の状態で受信側に転送している．

slsocket API 適用時において非適用時と比較すると CPU 使用率の変動が大きいが，これはサービス広告におけるブロードキャストを 5 秒間隔で送信している影響である．また，64 バイトの slsocket API 適用時における転送で CPU 使用率が低くなっている．通常，ミドルウェアの導入はコンテキストスイッチによる負荷を伴い CPU 使用率の上昇を招く．しかし, 送信側の CPU 使用率が常に 100% の状態であり, slsocket API 適用時の受信トラフィックが非適用時の受信トラフィックと比較して低くなったため，このような結果となった．

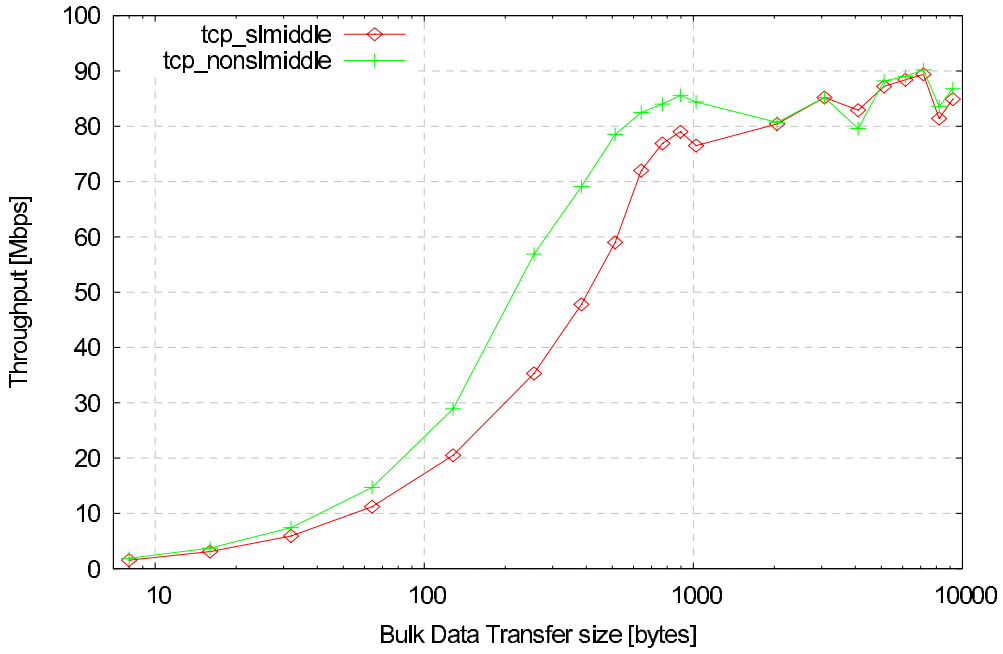


図 3.5.1 トランスポート層に TCP を用いたときのスループット

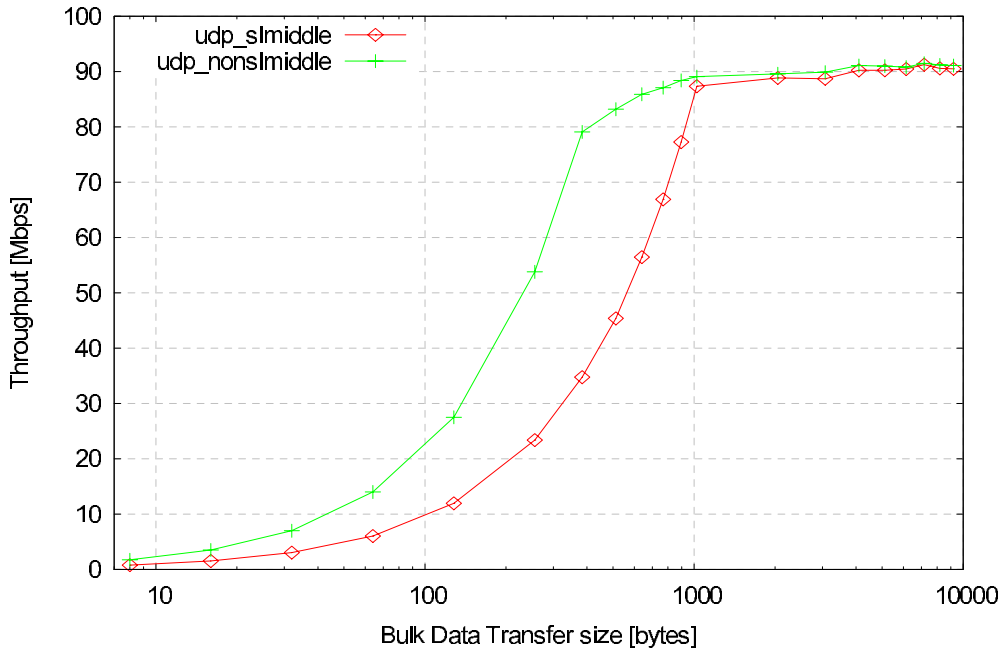


図 3.5.2 トランスポート層に UDP を用いたときのスループット

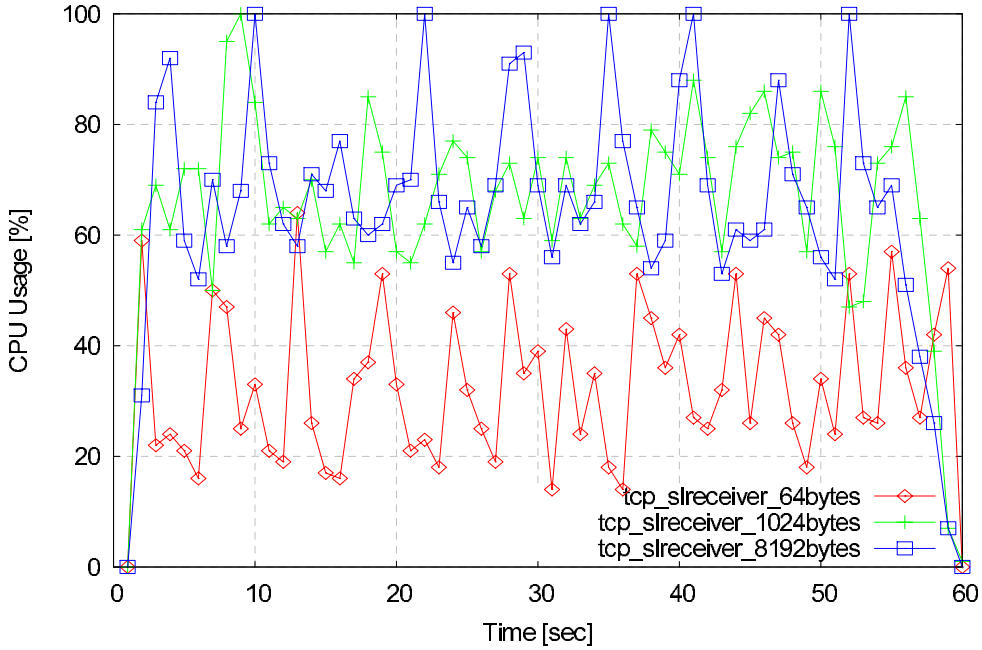


図 3.5.3 セッション層利用時の CPU 使用率変化

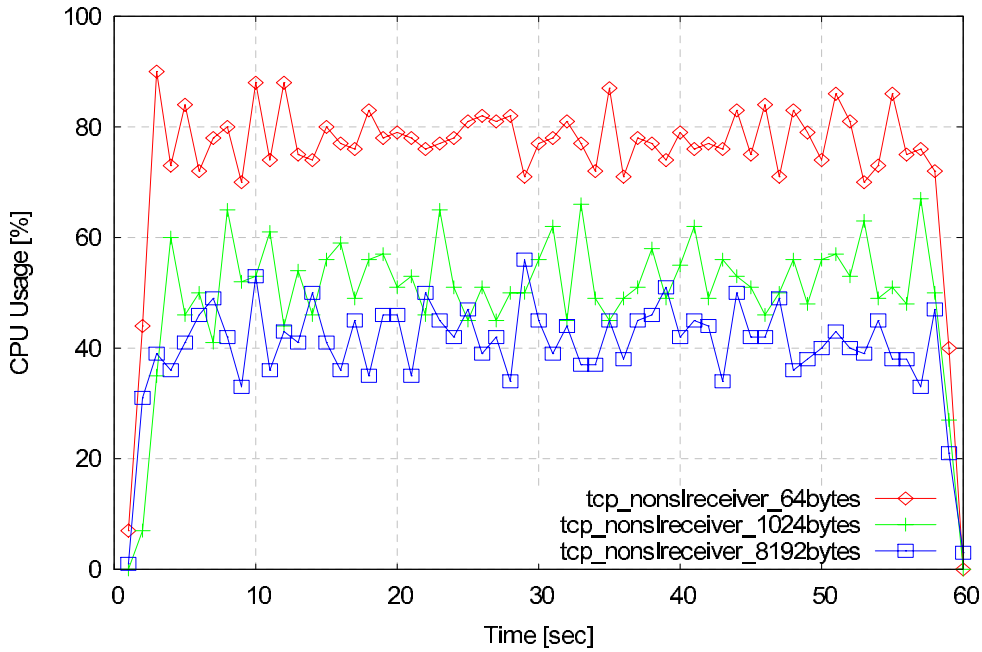


図 3.5.4 セッション層非利用時の CPU 使用率変化

3.6 関連研究

本節では，" セッション層 " に関係する技術および研究を示し，筆者らのセッション層との違いを明確にする．

ISO の Open Systems Interconnection (OSI) 7 階層通信モデル [13] には，" セッション層 " の機能が二つ示されている．一つは "session administration service" であり，二つのプレゼンテーション層のエンティティを " セッション層 " が関連づける機能である．もう一つは "session dialogue service" であり，接続したプレゼンテーション層エンティティ間のデータ交換の制御とデータ境界の制御，データ操作の同期制御を行う機能である．本論文で述べたセッション層は，OSI における "session administration service" を担うものである．

冒頭で述べたようにインターネットでは " セッション層 " が定義されていないが，" セッション層 " の機能は主に TCP 接続の解除, 切断に対応する手段として利用, 研究されている．

これまでの技術と筆者らのセッション層の最も大きな違いは，"session administration service" を実現するためにアプリケーションプログラムが通信に関する情報を保持しているか否かにある．すなわち，これまでの " セッション層 " の技術は，アプリケーションプログラムが通信相手に関する情報を DNS[14] や URL[15，16，17]，独自の名前空間を用いて定義しているのに対し，筆者らのセッション層ではユーザ主導型通信の実現のためにアプリケーションプログラムがこれらの情報を一切持たない．したがって，これまでの技術ではアプリケーションプログラムから通信が開始されるが，筆者らのセッション層ではセッション層制御部によって通信は開始され，アプリケーションプログラムはセッション層ミドルウエアからの接続を受け付けることで始めて通信することができるようになる．

現在のインターネットで最もよく利用されている " セッション層 " の機能は，HTTP の cookie である．HTTP の cookie は，WWW クライアントおよび WWW サーバが DNS 名や IP アドレスに関連づけて通信状態を保持しており，この情報を TCP 接続毎に送受信することで，過去の TCP 接続と現在の TCP 接続の継続性を仮想的に提供している．

モビリティサポートを実現するために " セッション層 " を導入する研究として Session Layer Mobility（SLM）[17] と Migrate[18] がある．SLM はユーザロケーションサーバを用いた新しい名前空間を利用しており，Migrate は DNS 名などの何らかの名前空間を用いることを前提にしている．また，Migrate は TCP にしか対応しておらずトランスポート層プロトコルに強く依存している．

VoIP の普及に伴って注目を集めるようになった Session Initiation Protocol（SIP）[15] は，" セッション層 " の導入を明確に示していないが，SIP URL を導入することでユーザの位置に依存しないサービスを実現しようとするものであり通信の接続解除を上位層で行うプロトコルのひとつである．SIP はアプリケーションプログラムが SIP URL を保持しアプリケーションプログラムから通信を開始するため本研究とは異なる．また，SIP は遠隔会議におけるマルチキャストアドレスの広告を起源とするため，広告されたアドレス，ポート番号に接続を試みることができるだけで，通信チャネルを積極的に制御する機構を持っていない．

3.7 むすび

本論文では，ユーザ主導型通信を実現するセッション層の設計と実装および評価について述べた．ユーザ主導型通信により，インターネットにおいて，伝達する情報の種類によらない普遍的な情報流通のフレームワークと通信相手との認証および安全な通信を保証するセキュリティフレームワークを兼ね備えた通信アーキテクチャを実現することができる．

参考文献

[1] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," ACM Transactions on Computer Systems, pp.277-288, 1984.

[2] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in Proceedings of the 17th ACM Symposium on Operating Systems Principles, pp.186 - 201, December 1999.

[3] 南正輝 , 森川博之 , 青山友紀 , " ユビキタス環境におけるサービス合成支援のためのインタフェース指向ネームサービス ," 電子情報通信学会論文誌 , Vol.J86-B, No.5, pp.777-789, 2003.

[4] I. Stoica, D. Adkins, S. Ratnasamy, S. Shenker, S. Surana, and S. Zhuang, "Internet indirection infrastructure," in Proceedings of the 1st International Workshop on Peer-to-Peer Systems , March 2002.

[5] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee, "Hypertext transfer protocol — HTTP/1.1," RFC 2616, Internet Engineering Task Force, June 1999.

[6] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in Proceedings of ACM Mobicom, November 1995.

[7] H. Elaarag, "Improving TCP performance over mobile networks," ACM Computing Surveys, Vol. 34 , Issue 3, pp.357-374, September 2002.

[8] B. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview", RFC 1633, Internet Engineering Task Force, June 1994.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services", RFC 2475, Internet Engineering Task Force, December 1998.

[10] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, "The design and implementation of the 4.4BSD operating system," Addison Wesley, April 1996.

[11] A. C. Snoeren, "A session-based architecture for Internet mobility," PhD thesis, Massachusetts Institute of Technology, December 2002.

[12] CRN Monitor, "http://www.vector.co.jp/soft/win95/hardware/se331669.html."

[13] H. Zimmerman, "OSI reference model — the ISO model of architecture for open system intereconnection," IEEE Transactions on Communications, COM Vol.28, No.4 pp.425 - 432, April 1980.

[14] P. Mockapetris and K. Dunlap, "Development of the domain name system," in Proceedings of ACM SIGCOMM, pp.123-133, April 1988.

[15] J. Rosenberg, H. Schulzrinne, G. Caramillo, A. Jonston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. "SIP: Session initiation protocol," RFC 3261, Internet Engineering Task Force, June 2002.

[16] R. Moskowitz, "Host identity payload and protocol," Internet Draft, Internet Engineering Task Force, October 2001. draft-moskowitz-hip-05.txt (expired).

[17] B. Landfeldt, T. Larsson, Y. Ismailov, and A. Seneviratne, "SLM, a framework for session layer mobility management," in Proceedings of IEEE International Conference on Computer Communications and Networks, pp.452 - 456, October 1999.

[18] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp.155 - 166, Boston, Massachusetts, August 2000.

4 セッション層における鍵管理

4.1 はじめに

セッション層は，アプリケーションプログラムが通信に関する情報を保持せず，セッション層制御部が管理している．そのため，セッション層制御部が通信相手と利用 IP アドレスなどの通信に関する情報を交換し，ミドルウエアが利用する IP アドレスを変更するだけでモビリティサポートを容易に実現することができる．

本章では, 単一の端末がネットワーク間を移動するターミナルモビリティだけでなく, ユーザが利用するアプリケーションプログラムを通信の途中で切り替えるサービスモビリティも含めたモビリティサポートの実現を考慮したセッション層における鍵管理手法について述べる．サービスモビリティでは，アプリケーションプログラムが動作している端末が変更になる場合があるため，セッション層制御部間の認証技術が重要な課題となる．

以下では，4.2 でサービスモビリティについて述べ，4.3 でセッション層を用いたモビリティサポートの概要を示す. 次に, 4.4 でサービスモビリティに必要な認証技術を述べる. 4.5 でサービスモビリティに必要な認証を実現する技術として，key-insulated 公開鍵暗号方式について述べる．そして，4.6 でセッション層における鍵管理手法として，key-insulated 公開鍵を用いたセッション層モビリティサポートについて述べる．

4.2 サービスモビリティ

本節では, まずサービスモビリティの概要を示し, これまで主に開発が進められてきたターミナルモビリティとの違いを明らかにする.

4.2.1 サービスモビリティ

インターネットは，接続されるコンピューティングデバイスやそれらをインターネットに接続するアクセスリンクにおいて多様化の時代を迎えている．リソースが多様化したインターネット環境では，現在の携帯電話端末のように単一のデバイスを常時持ち歩き使い続けるだけではなく，その場その時の利用要求に応じてリソースを周辺環境の中から選択し切り替えて利用することが望まれる．すなわち，ユーザの周辺環境やリソースに関する要求事項は時々刻々と変化するものであり，ユーザは常に最適な通信環境を利用するために，通信中であっても積極的にリソースの選択・切り替えを可能にする技術（サービスモビリティ）が必要となる．以下に簡単なサービスシナリオを述べる（図 4.2.1）.

ここはアリスのオフィスである．商品の買い付けに出かけている部下のボブからアリスに電話がかかってきた．ボブは，買い付けの是非についてテレビ電話で商品を見せながら，アリスの意見を参考にしたいと考えたのである．しかし，彼女は手近にあった携帯電話でボブの電話にでてしまった. ボブはアリスに電話の意図を伝え, アリスは携帯電話からデスクトップコンピュータに通信を切り替えた．アリスは，大画面のディスプレイが接続されたデスクトップコンピュータで，その商品を細部まで確認し，ボブに購入を勧めた.

サービスモビリティは，ユーザが利用したいと思うデバイスを周辺環境から発見し選択するサービス選択技術，サービス選択によって決定したデバイスに通信を移動させるサービス移動技術，新しく利用するデバイスや通信リンクの性能に応じてサービス品質を柔軟に変化させるサービス適合技術によって構成される．上記のシナリオを例にとると，アリスが商品を確認する際にオフィスの中から大画面のディスプレイが接続されたデスクトップコンピュータを選択可能にする技術がサービス選択であり，その後，アリスがボブとの通信を携帯電話からデスクトップコンピュータへの切り替えられるようにする技術がサービス移動技術，そして，音声通信に映像通信を加えるともに，商品の細部を確認するために高解像度の映像に通信を変更させる技術がサービス適合技術となる．本論文では，特に，通信サービスを継続しながら，通信のエンドポイントをユーザの要求に応じてネットワークや端末を跨いで切り替えるサービス移動技術について述べる.

4.2.2 サービスモビリティとターミナルモビリティ

これまでインターネットにおける通信の移動透過性は，移動に無関係な識別子を設け，その識別子に対して移動によって変化する識別子を対応づけることによって実現されてきた.

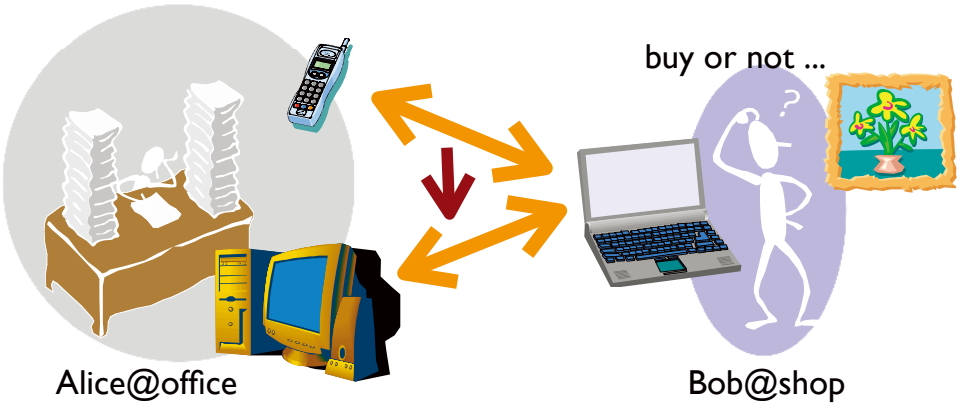


図 4.2.1 サービスモビリティシナリオ

さらに，この対応付けの変更を行う際に認証処理を行うことで，モビリティサポートが持つ潜在的なセキュリティ上の脆弱性である通信ハイジャックを防いでいる．IETF を中心に検討されてきた Mobile IP [1] は，アドレス空間の異なるネットワークを渡り歩く単一の端末に対して，パケット到達性を与える技術であり，ターミナルモビリティサポートと呼ばれている．ターミナルモビリティでは，端末に付与される IP アドレスは移動によって変化するものの，端末自体は変化しないため，端末ごとに固有の識別子を与えることで移動に無関係な識別子を導入することが可能となる．また，対応付けの変更に必要な認証情報はユーザが利用する端末に保存されており，その端末の安全性が確保されていれば安全に識別子の対応付けを変更できる．

例えば，Mobile IP や LIN6 [2] では端末ごとの固有識別子としてネットワークアドレスと同じ空間に端末のネットワーク位置に依存しないアドレス（順に，ホームアドレス・LIN6 アドレス）を定義している．また，TCP Migrate [3] は問題点を TCP に絞り，端末が不変であれば IP アドレスの変化のみでポート番号やシーケンス番号など TCP 固有の情報に依存しないことから，TCP を拡張し TCP 内部に TCP コネクション固有の識別子を設けることで移動による IP の変化に対応している．すなわち，ターミナルモビリティでは端末が不変であるという制約条件を生かし，移動に無関係な識別子を導入しているといえよう．また，認証に関しても同様に，認証情報の安全性を端末の単一性が保証している．

一方で，サービスモビリティは端末を跨いだ移動を実現するため，端末の固有情報や TCP コネクションの固有情報を移動に無関係な識別子として用いることはできない．さらに端末が変わるため，通信を識別するための情報，および対応付けを変更する際に必要な認証情報を端末から端末に移動させる必要がある．前者に関して，筆者らは端末や TCP コネクションなどの固有情報に依存しない自由度の高い通信の実現に向けてセッションレイヤモビリティサポートを検討してきた．そこで本稿では，特にサービスモビリティにおける移動認証技術に着目する．サービスモビリティでは，これまでのネットワーク移動のみを対象にしたターミナルモビリティと異なり，端末を跨いだ移動となるため，認証情報の安全性を端末の単一性に求めることができない．すなわち，サービスモビリティを実現するためには，複数の端末を切り替えながら利用する場合においても認証情報を安全に管理するとともに，セキュアなサービスモビリティを実現する手法について新たに検討する必要がある．以下では，基盤となる移動透過技術としてセッションレイヤモビリティサポートを採用し，公開鍵を変更することなく秘密鍵を更新可能な key-insulated 公開鍵暗号方式 [4] を用いたセキュアなサービス移動技術の実現について述べる．

4.3 セッション層を用いたモビリティサポート

セッション層を用いたモビリティサポートはトランスポート層とアプリケーション層の中間に設けられたセッション層によってモビリティをサポートする．具体的には，セッション層がアプリケーションプログラムに対して端末や IP アドレス等の端末固有情報に依存しない通信インタフェースを提供し，セッション層ミドルウエアが IP アドレスの変更に対応することでモビリティをサポートする（図 4.3.1）．

まず，現在最もよく利用されている通信インタフェースである socket API [5] のモビリティにおける問題点について述べる．socket API は，デバイス I/O と通信 I/O を統一的に扱うことができる優れた I/O フレームワークであり，アプリケーションプログラムはこの socket インタフェースを直接呼び出して使っている．しかしながら，socket API は，扱う I/O が静的であることを前提としており，I/O が動的に変化することを想定していない．そのため，例えば IP アドレスの変更といった I/O に動的な変化が生じた場合，アプリケーションプログラムは，その通信インタフェースを利用することができなくなり，通信は途絶してしまう．これは，通信インタフェースに IP アドレスなどの端末固有情報が強固に関連づけられ，通信インタフェースが開かれている限り関連づけられた情報を変更できないことが原因となっている．したがって，アプリケーションプログラムが通信を維持したまま，コンピューティングデバイスや通信リンクなど通信のエンドポイントを柔軟かつ積極的に切り替えられるようにすることは，このような静的な I/O を前提とし，それをアプリケーションプログラムが直接利用する通信 API では困難である．

一方 slsocket API は，アプリケーションプログラムが扱う通信インタフェースから通信相手情報や端末固有情報を廃しており，対向となる slsocket を保持するセッション層ミドルウエアが socket API を用いた静的なインタフェースを構築し slsocket と接続している．slsocket API を用いた通信ではアプリケーションプログラムが通信相手情報を持たないため，アプリケーションプログラムから通信が開始されるのではなく，セッション層制御部からの指示に基づいてセッション層ミドルウエアがアプリケーションプログラムに通信接続を行うことで通信が開始される．

セッション層モビリティサポートは，アプリケーションプログラムや slsocket API を改変することなく，セッション層ミドルウエアがモビリティ時に socket API を用いて通信インタフェースを作成し，新しい通信インターフェースを slsocket に接続する機構である．セッション層モビリティサポートでは，セッション層ミドルウエアがアプリケーションプログラムに対して通信接続を行うため．アプリケーションプログラムの切り替えも容易であり，ネットワークを跨ぐモビリティだけではなく，アプリケーションプログラムを切り替えるモビリティが実現可能である．

なお，セッション層では，セッション層制御部が通信相手のセッション層制御部と認証，通信情報の交換を行っており，セッション層モビリティサポートにおいてもセッション層制御部が通信相手のセッション層制御部と通信情報の変更に関するやりとりを行う．このやり

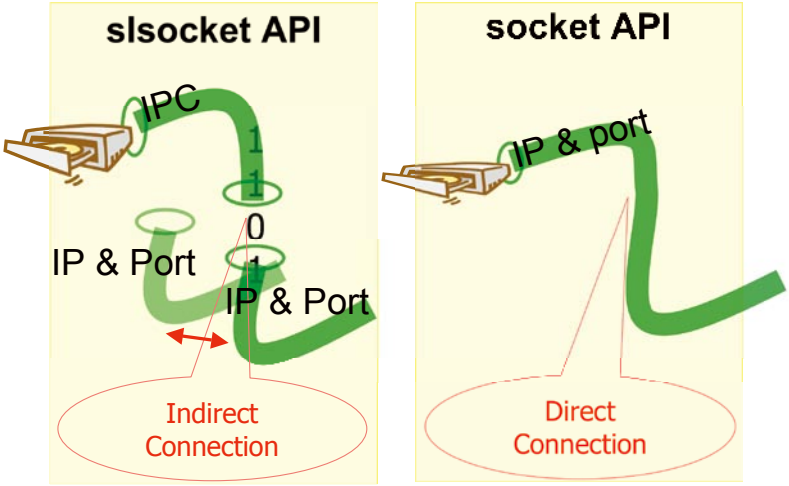


図 4.3.1 セッション層を用いたモビリティサポート

とりは，一連の通信チャネルを識別する情報（セッション情報）を用いて行う．

4.4 サービスモビリティに必要な認証技術

サービスモビリティを安全に利用できるようにするためには単純な切替処理だけでは十分ではない．なぜなら，モビリティサポートは宛先の動的な切り替えを可能にする技術であるため，悪意のあるユーザによって意図せず通信を切断されるだけでなくハイジャックされる可能性を潜在的に持っているからである．

まず，サービスモビリティにおけるセキュリティ上の脆弱性について述べる．セッション層モビリティサポートでは，セッション情報を通信のエンドポイントとなる端末から端末へと受け渡していくことで,モビリティを実現している．そのため,認証情報を含んだセッション情報が移動後も端末に残り，その情報を悪意のあるユーザに盗まれると，ユーザの意図しない移動要求を送られ，セッションが乗っ取られる可能性がある．特にコンピューティングデバイスが遍在する環境では，ユーザは，持ち歩いているデバイスだけでなく周囲のデバイスも自由かつ積極的に利用すると考えられるため，常に安全性が保証された端末を利用できるとは限らない．このような環境におけるモビリティサポートでは，ユーザの利用端末は安全ではないという前提に立って認証処理を設計する必要がある．

次に，セッション層技術はエンドツーエンドの考え方に立脚しているため，認証処理は当該通信のもう片方のエンドポイントである通信相手も自由にデバイスやネットワークを選択できなければならない，したがって，セッション層制御部が通信情報の交換を行う相手端末は，通信によって毎回変化する可能性がある．このような環境では，秘密情報の共有（共通鍵暗号方式)を前提にすることはできず,公開鍵暗号方式を用いることが妥当であろう.また，ユーザのどちらかが端末を変更するごとに鍵を再生成するという手法も考えられるが，鍵生成に要する計算能力や公開鍵の交換手段をモバイル環境における貧弱な計算機環境から考えると非現実的であり，仮にこのような手法を採用したとしてもその高い認証コストから移動を自由に行えなくなる．

そして，モビリティサポートそのものが持つ脆弱性を回避する機構を検討しなくてはならない．すなわち，移動の切り替えは通信相手のセッション層が通知するメッセージによって行われるため，送られてきた移動要求メッセージがユーザの意図したものであるか，移動要求メッセージを送信したとしてその内容は改竄されていないかを，移動要求メッセージの受信時に確認しなければならない．このような確認には，メッセージの送信者を特定し，内容改竄の有無を検出できる電子署名が必要である．また，悪意のあるユーザが過去の移動要求メッセージをコピーして保存し，その後再送信するリプレイアタックを防ぐことも必要となる．この場合，メッセージそのものは送信者が電子署名を付与しているため，何らかのシーケンス番号によって管理する必要がある．

以上から，セッション層モビリティサポートに要求される認証処理は，下記の要求条件を満たす必要がある．

(1) 複数の端末を切り替えて利用すること（ただし，同時利用は 1 端末のみ）

(2) 利用端末は，安全ではないという前提をもつセキュリティモデルであること

(3) 認証情報を端末間移動させる際に，高い計算能力，複雑な鍵交換のシステムを必要としないこと
(4) 移動要求メッセージの内容および送信者を保証する電子署名を利用可能であること
(5) リプレイアタックを防止するシーケンス番号が与えられていること

上記条件を満たす暗号方式として，次節で述べる key-insulated 公開鍵暗号方式が存在する．

4.5 Key-insulated 公開鍵暗号方式

Key-insulated 公開鍵暗号方式 [7] は，2002 年に Y. Dodis らによって提案された公開鍵暗号方式である．この暗号方式は，秘密鍵の漏洩が現在のセキュリティシステムの脆弱性に繋がっていることを問題点とし，これまで，安全性が確保された単一のデバイスで管理することを前提としてきた秘密鍵にステージと呼ばれる期間を与えることで，安全性が確保されない端末においても秘密鍵を利用可能とするものである．（図 4.5.1）仮に特定のステージの秘密鍵が漏洩したとしても，ステージを変化させれば安全性を確保することが可能になる．また，この暗号方式において公開鍵はステージに依らず不変である．以下に，この暗号方式の概要を簡単に示す．なお，本暗号方式は，暗号化・復号化，電子署名のいずれにも用いることができる．

ユーザは，ひとつの公開鍵 PK を決定する．この PK に対応する親秘密鍵 SK* を物理的に安全なデバイスに保管する．全ての復号処理は，秘密鍵の漏洩が心配される安全性が保証されないデバイスにおいて行うものとする．この暗号方式は，ステージ 1，2，. . . , N（単純にこれをある単位時間と考えてもよい；例，一日）に分けて処理する．それぞれのステージの最初に，ユーザは，安全なデバイスから復号に使う一時的な秘密鍵を受け取る．ここで，ステージ i における一時的な秘密鍵を SKi とする．一方，公開鍵 PK は，メッセージを暗号化するのに使われ，ステージに依らず不変である．その代わり，暗号文には，ステージを示すラベルが付与される．また，復号処置を行う非安全デバイスは，繰り返しによる鍵の漏洩の脆弱性を持っているため，最大利用ステージ t（t < N）を設け，で用いることを前提とする．

具体的な SKi の生成方法は以下の通りである．まず，（PK，SK）を生成し，PK を公開し，さらに SK から SK* と SK0 を生成する．SK0 は，ユーザがステージ 0 における秘密鍵として利用し，SK* は安全なデバイスに保管する．次にユーザがステージ i における秘密鍵を要求すると，SK* が保存されているデバイスは，部分鍵 SKi' をユーザに渡す．ユーザは，SKi-1 と SKi' を用いて完全鍵 SKi を生成する．この方法を用いると全てのステージにおいて，SK* だけ，もしくは SKi-1 だけではステージ i の完全鍵を生成できないため，特定の端末に依存しない鍵の保管が可能になり，セキュリティが高まる．

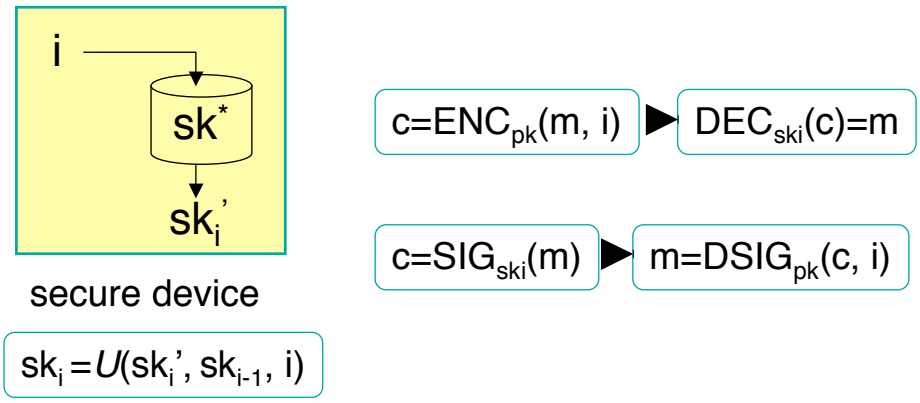


図 4.5.1 Key-insulated 公開鍵暗号方式

4.6 セッション層における鍵管理方式

本節では，セッション層における鍵管理方式の一例として，Key-insulated 公開鍵暗号方式を用いたモビリティサポートを示す．

ここでは，一つのセッションに対して一つの key-insulated 公開鍵を用いるものとする．これは，一人のユーザが同時に複数のセッションを別々の端末で動作させることが考えられるためである．

まず，key-insulated 公開鍵暗号方式が前節で述べた要求条件を満たしていることを示す．要求条件 1，２および 4 に関しては，4.5 に述べたとおりであるため省略する．要求条件 3 に関して，一般的な公開鍵暗号方式の鍵生成にかかる計算量は O(logN) であるのに対し，key-insulated 公開鍵暗号方式の SKi 生成にかかる計算量は O(1) であり，計算量は少ないため条件を満たしている．また，要求条件 5 については，key-insulated 公開鍵暗号方式におけるステージ i を移動要求の回数に対応させることで，リプレイアタックを防止することができる．これは，key-insulated 秘密鍵 SKi がステージ i においてのみ有効であることを用いている．移動要求が受理されたときに，ステージを更新し，新たな秘密鍵 SKi+1 を作成することにより SKi を移動要求メッセージの秘密鍵として利用することを無効化することができる．

以下では，セッション層モビリティサポートへの key-insulated 公開鍵暗号方式の適用に関して具体的な手法を述べる．まず通信開始時における鍵交換の仕組みについて述べ，次に移動要求時の処理について述べる．

通信開始時の鍵交換

Key-insulated 公開鍵は，前に述べたように一セッションに対して一つの公開鍵を割り当てる．すなわち，通信開始時に当該セッションで利用する key-insulated セッション公開鍵を交換しなければならない．公開鍵暗号方式では，公開鍵を安全に交換することができなければ，中間者攻撃によって通信の安全性を確保することができない．ここでは，PKI（Public Key Infrastructure）を用いることで，セッション公開鍵の安全な交換を実現する．前提として，全てのユーザは PKI に公開鍵を預けており，PKI を通して，任意のユーザの公開鍵を安全に入手できるものとする．ユーザ A は，ユーザ B と通信を行おうとしているものとする．なお，安全なデバイスにセッション親秘密鍵 SSK* が保存されており，利用端末には，ステージ 0 におけるセッション秘密鍵が保存されているものとする（図 4.6.1）．図中の "E_" は暗号処理を，"D_" は復号処理を示す．

A は B の公開鍵を PKI を通して入手し，そのユーザ公開鍵 UPKB を用いて，セッション用の公開鍵 SPKA および乱数 A を暗号化して B に送信する．B は，A からのメッセージを復号し，SPKA と乱数 A' を得る．B は続いて A のユーザ公開鍵 UPKA を PKI を通じて入手し，B のセッション用公開鍵 SPKB と，乱数 B，乱数 A' を UPKA を用いて暗号化し A に返信する．A は，B からのメッセージを復号し，SPKB，乱数 B'，乱数 A' を得る．ここで，最初に生成

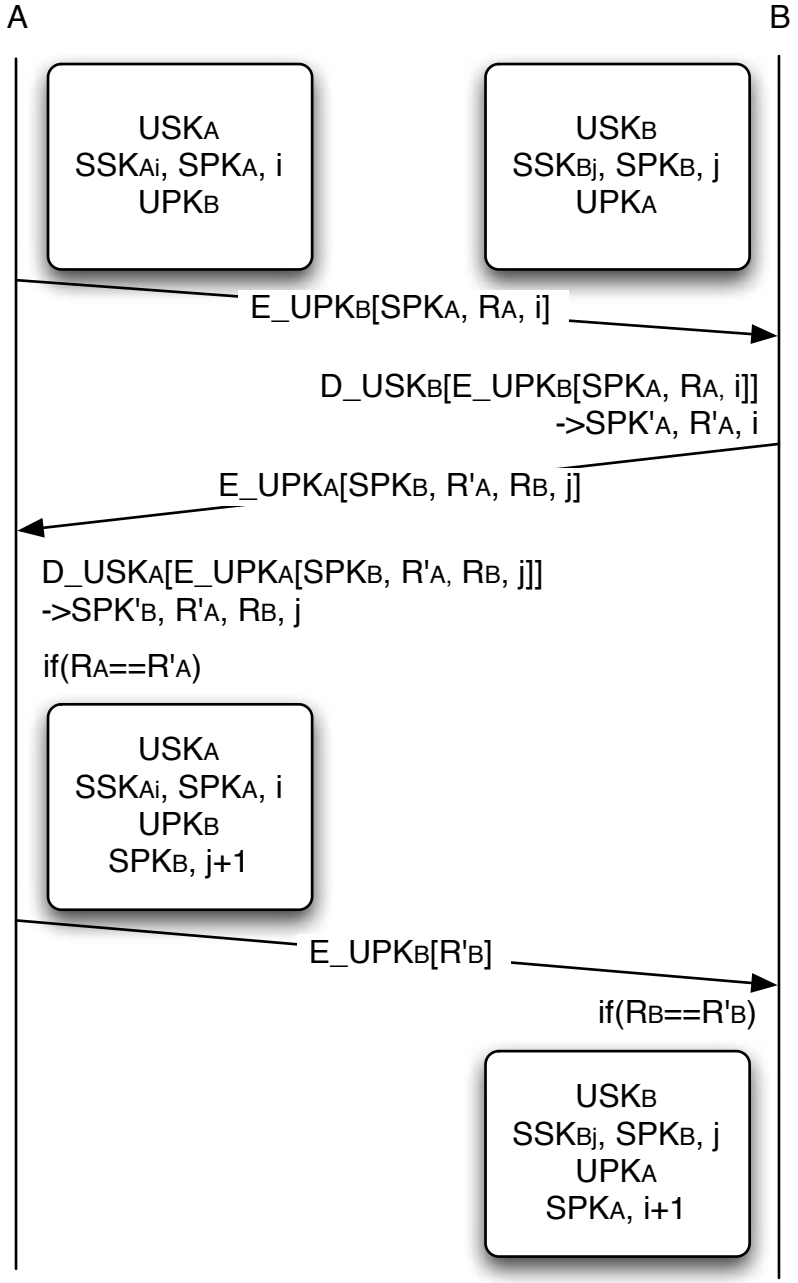


図 4.6.1 通信開始時の鍵交換シーケンス

した乱数 A と乱数 A' の一致を確認し，SPKB を保存する．最後に，UPKB で乱数 B' を暗号化して送信し, B においても乱数 B と B' の一致を確認し, SPKA を保存する. 鍵交換終了後, A, B の利用端末では，移動要求メッセージの処理に備える．移動の度にステージを変えることで，本システムは安全性を確保しているため，移動要求メッセージの待ち受けは，ステージ 1 で待つことになる．ただし，鍵交換終了後で移動を行うまでの通信には，ステージ 0 における SSK0 が用いられることに注意されたい．

ここでは, key-insulated 公開鍵における最初のステージを 0 としているが, 任意のステージ i, j から始めることが可能である. この場合, 利用端末に SSKi, SSKj が保存されており, セッション公開鍵の交換においてステージ i，j をそれぞれ通知することになる．なお，i と j はそれぞれ独自のパラメータであり，A の移動回数に応じて i が増加し，B の移動回数に応じて j が増加する．詳細は，次に述べる．

移動要求時の処理

移動要求時には，新しいステージのセッション用秘密鍵を用いて移動を相手に通知する．ここでは，A が端末 A1 から A2 に移動するときを例に述べる．本手法では，移動先端末 A2 から移動要求メッセージを送信する．これは，特にモバイル環境において，不意にネットワークから切断される等のディスコネクション状態が発生しても，新しい端末で通信を継続できるようにするためである．まず, A は, 移動先端末 A2 に新しいステージ i のセッション用秘密鍵を保存する．（注:A1 には, ステージ i-1 のセッション用秘密鍵が存在している.）（図 4.6.2）図中の "S_" は電子署名の処理を，"V_" は電子署名の認証処理を示している．

A2 は, SSKAi を用いて移動要求に関する電子署名を作成し, これとステージを示す i をメッセージに付与して，B に送信する．B は，ステージ i-1 の移動要求メッセージ受理後，ステージ i の移動要求メッセージを待っており, A2 からのメッセージを認証する. 認証処理後, B は, A2 に移動要求に対する ACK を SSKA を用いて署名を付けて返す．その後，B は，A1 に対し, 接続終了要求を SSKA を用いて署名を付けて送信する. これらふたつの B からのメッセージは，ステージ j のセッション秘密鍵 SSKBj を用いて電子署名が付与されている．A1 および A2 で移動要求待ち受け用のステージは, j+1 であるが, この処理は移動要求ではないため, 待ち受け用ステージ j+1 よりステージ数が一つ少ない，ステージ j の鍵を用いて処理されている．なお，ステージ j の電子署名を送信しても，key-insulated 公開鍵では公開鍵が不変なため，認証することは可能である．最後に，B は移動要求を受け付けるステージを i+1 にする．なお，図中の session ID は，セッション認証情報の検索を高速に行うためのものである．

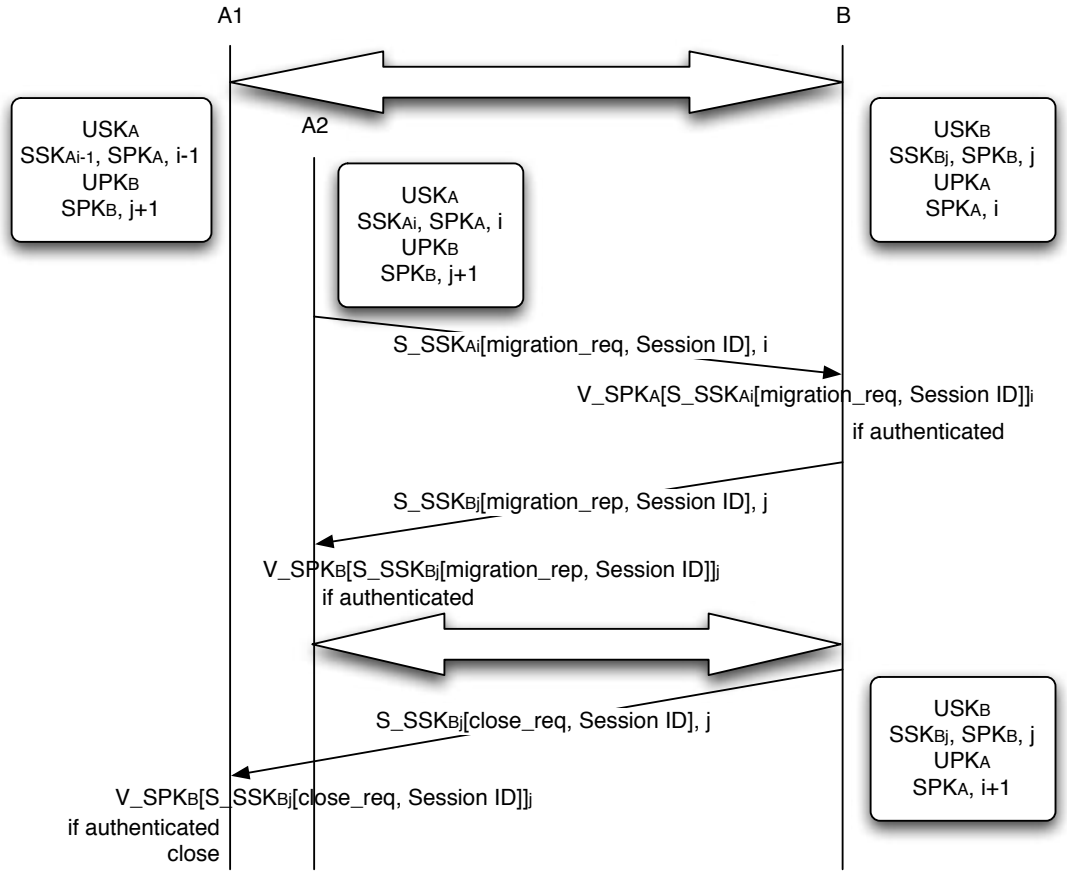


図 4.6.2 通信制御情報の移動時のシーケンス

4.7 むすび

本章では，セッション層における鍵管理方式について述べた．特に，サービスモビリティを実現するためのサービス移動をセキュアに行うことを想定し，具体的な手法としてkey-insulated 公開鍵暗号方式を用いたセッション層モビリティサポートについて述べた．Key_insulated 公開鍵暗号方式は，秘密鍵を安全でないデバイス上で利用することを想定し，秘密鍵がステージに分けて管理されている．本方式では，ひとつのセッションに一つのkey-insulated 公開鍵を割り当て，ユーザの移動にステージを対応づけることにより，デバイスに依存しないモビリティサポートの認証処理を実現することができる．

参考文献

[1] C. E. Perkins, "IP mobility support for IPv4," RFC 3220, Internet Engineering Task Force, January 2002.

[2] M. Kunishi, M. Ishiyama, K. Uehara, H. Esaki, and F. Teraoka, "LIN6: A new approach to mobility support in IPv6," In Proceedings of the 3rd International Symposium on Wireless Personal Multimedia COmmunications (WPMC), pp.1079-1084, November 2000.

[3] A. C. Snoeren and Hari Balakrishnan, "An end-to-end approach to host mobility," In Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 155-166, Boston, Massachusetts, Aug. 2000.

[4] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated Public-key Cryptosystems," Advances in Cryptology -EUROCRYPT 2002, Lecture Notes in Computer Science vol. 2332, L. Knudsen ed., Springer-Verlag, 2002.

[5] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, "The Design and Implementation of the 4.4BSD Operating System," Addison Wesley, Reading, Massachusetts, April. 1996.

5 セッション層が管理する情報
–省電力モバイル端末の例–

5.1 はじめに

セッション層は通信に必要な情報を管理機構である．本章では，セッション層が具体的にどのような通信情報を管理するのかについて，IEEE 802.11 省電力モバイル端末を例に述べる．

セッション層が通信チャネルを構築するのに必要な最小限の情報は，IP アドレス，ポート番号，トランスポート層プロトコルである．以下では，IEEE 802.11 省電力モバイル端末の TCP スループットの改善方法を示し，トランスポート層の内部情報まで管理した場合のシステムに与える影響を考察する．

本章の構成は以下の通りである．まず, 5.2 で IEEE 802.11 省電力モードについて説明し，5.3 で IEEE 802.11 における省電力モードと省電力モードにおける TCP スループットの劣化について説明する．次に，5.4 で省電力モバイル端末の TCP スループットの改善手法の詳細を述べ，5.5 で本手法の実装と性能評価を示す．

5.2 IEEE 802.11 省電力モード

近年，IEEE 802.11 [1] に基づいた無線ネットワーキングが急速に普及している．IEEE 802.11 無線インタフェースを保持する端末は，2004 年に発売された全ノート型コンピュータの 70% に達する [2] といわれている．また, IEEE 802.11 基地局は, 当初オフィスや自宅, 喫茶店などスポット的に設置されてきたが，最近になってサービスエリアの面的な展開も進みつつある [3]．このように IEEE 802.11 の通信基盤は整いつつあるが，長時間の無線通信を可能にする省電力技術がなければ，電源の確保が問題となりモバイルネットワーキングを実現することができない．

特に，IEEE 802.11 は CSMA/CA で動作するためモバイル端末が常に待ち受け状態になければならず，送受信を行わない待機時であっても電力を供給しなければならないため電力消費が激しい．例えば，ルーセントテクノロジの PCMCIA 型無線 LAN カードでは，送受信時の消費電力がそれぞれ 1.65W と 1.40W であるのに対し待機時の消費電力が 1.15W であり [4]，送受信時に匹敵する電力を消費していることが分かる．

IEEE 802.11 では省電力無線通信を実現するためのプロトコルとして省電力モード（Power Saving Mode）を標準化している．省電力モードは，基地局とモバイル端末が連携し，基地局が省電力モードのモバイル端末へのフレームの送信開始タイミングを 100 ミリ秒間隔にすることで，端末のスリープ時間を確保する技術である．上述の無線 LAN カードにおけるスリープ時の消費電力は 0.045W であり，省電力モードを用いることで消費電力を大幅に削減することが可能になる．

しかしながら，省電力モードを利用すると TCP 通信のスループットが大幅に低下する．これは，基地局がモバイル端末宛のフレームを一時的に蓄積するため TCP の ACK 応答も遅れ，固定端末の送信ウィンドウが十分に開かないことによる．

これまで，無線 TCP スループットの性能改善に関する研究は数多く行われてきた [5, 6]．これらの研究は無線区間でのパケットロスに起因するスループット劣化を扱っていたためパケットの再送を効率的に行うことが重要であった．これに対し，本論文で述べる省電力モードにおける TCP スループットの劣化は過度に長くなる通信遅延時間に起因しており，長い通信遅延時間により ACK が遅れても TCP の送信ウィンドウを開く方法を検討する必要がある．省電力モードにおける TCP スループットの改善に取り組んだ研究として，BSD [7] がある．BSD は，現在のアプリケーションサービスのほとんどが WWW 閲覧であることに注目し，データ取得中はスループットの低下を防ぐために省電力モードにならずデータ取得後のユーザのページ閲覧時間にスリープ動作を行う．具体的には，無線 LAN のドライバが通信状態を常に監視して省電力モードの利用タイミングを変更することで，TCP スループットの確保と省電力の両立を実現している．

筆者は，無線 LAN ドライバの機能拡張などモバイル端末に変更を加えずに，基地局による疑似 ACK 送信と基地局間連携によって，省電力モード時の TCP スループットの改善を図る．疑似 ACK 送信は，モバイル端末の TCP 接続を基地局が監視し，基地局が TCP の状

況にあわせて疑似 ACK を生成し固定端末に送信することで固定端末の送信ウィンドウを開き TCP スループットを確保する技術である．基地局間連携は，基地局の疑似 ACK 送信によって TCP のエンドツーエンドのセマンティクスがモバイル端末のハンドオーバ時に損なわれるのを防ぐために，モバイル端末が ACK を返していない TCP パケットをハンドオーバ前後の基地局間で受け渡す技術である．

5.3 省電力モードにおける TCP 性能

5.3.1 省電力モード

IEEE 802.11 では，基地局とモバイル端末が連携動作することでモバイル端末の消費電力を低減する省電力モードが規定されている．

IEEE 802.11 の省電力モードの動作概要を以下に示す．まず，省電力モードで動作するモバイル端末は基地局にアソシエーションするときに，省電力モードのフラグを立てて基地局に省電力モードで動作することを通知する．この通知を受けると，基地局は省電力モードのモバイル端末宛のフレームが届いた際に，そのままモバイル端末に転送するのではなく，当該フレームを基地局の送信キューに一時的に蓄積するよう動作を変更する．基地局は，標準的には 100 ミリ秒間隔の TIM（Traffic Indication Map）でビーコンを送信しており，このビーコンには省電力モードで動作するモバイル端末宛のフレームが基地局に蓄積されているかどうかの情報が記載されている．モバイル端末は，TIM に合わせてスリープ状態から復帰し，基地局からのビーコンを受信して，自端末宛のフレームが基地局に蓄積されているかどうかを確認する．未受信フレームが存在する場合，端末は基地局の送信キューに蓄積されたすべての未受信フレームの受信処理を行い，終了後再度スリープ状態になる．以上の処理により，省電力モードでは受信待機時間をスリープ時間に充てることが可能になり，消費電力を低減することが可能になる．なお，省電力モードであっても，モバイル端末がフレームを送信する場合は TIM を待つことなく CSMA/CA に従って送信処理が行われる．

5.3.2 省電力モードにおける TCP 性能

省電力モードにおける TCP スループットの低下の原因について述べる．省電力モードでは，モバイル端末宛のすべてのフレームが基地局で蓄積された後にモバイル端末に届く．基地局における蓄積時間はパケットの到着タイミングによって 0 から TIM 時間まで変化するため，モバイル端末に届くパケットはモバイル端末のスリープ時間に対応して通信遅延時間が増加しスループットが低下する（図 5.3.1）[7].

スループット低下の一例を示すと，通信遅延時間 10 ミリ秒の通信端末間でボトルネックリンクを IEEE 802.11b の無線区間とした環境で，500KB のファイルを TCP 転送した場合のスループットは，省電力モードを用いない場合 3.2Mbps であるのに対し省電力モードを用いる場合 1.6Mbps となる．

スループットへの影響は通信遅延時間に占める蓄積時間の割合によって異なる．蓄積時間を除いた通信遅延時間が非常に短い場合，モバイル端末の ACK に促されて固定端末が送信する TCP パケットは，モバイル端末がスリープ状態に遷移する前に基地局に届く．そのため，モバイル端末はスリープ状態にならずに受信を継続し，スループットに与える影響は小さい．通信遅延時間が長くなると，TCP パケットが届く前にモバイル端末がスリープ状態に遷移してしまうため，徐々にスループットへの影響は大きくなる．通信遅延時間がさらに長くな

ると，TCP の送信レートに対する通信遅延時間の影響が顕著になり，蓄積時間の影響が相対的に小さくなるためスループットへの影響は再び小さくなる傾向にある．

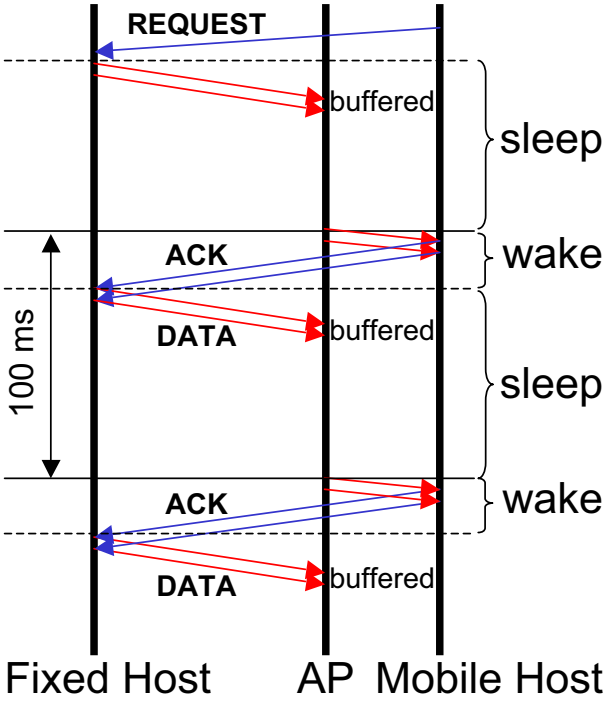


図 5.3.1 省電力モード時の TCP の挙動

5.4 省電力モバイル端末の TCP スループット改善手法

5.4.1 アプローチ

筆者は，広く普及しているモバイル端末や固定端末に手を加えることなく基地局だけに機能を追加して，省電力モードで動作するモバイル端末の TCP スループットの改善を図る．本手法の技術的特徴は，基地局の疑似 ACK 送信による TCP スループットの改善と，モバイル端末と固定端末間の TCP セマンティクスの保護である．基地局による疑似 ACK 送信はスループットの改善に寄与するが TCP のエンドツーエンドのセマンティクスを壊すため，TCP セマンティクスの保護が必要不可欠である．

ここで，筆者が実現する TCP のセマンティクスは，エンドツーエンドの流量制御や再送制御などのパケット単位のセマンティクスではなく，エンドツーエンドで信頼性のあるデータ伝送が完了したかどうかをモバイル端末と固定端末が TCP のコンテキストの中で把握できるフロー単位のセマンティクスである．実際, 現在のインターネットの階層化モデルでは，アプリケーションプログラムは，アプリケーションの独自プロトコルを用いない限り通信途中で通信相手へ届けられたデータ量を把握することはできず，TCP 接続が正常に終了することでデータ伝送の完了を確認しているにすぎない．したがって，上記のセマンティクスを保つことができれば実用上の問題は生じない．

・TCP スループットの改善

基地局がスリープ中のモバイル端末の代わりに固定端末に疑似 ACK を返すことで，省電力モードによる蓄積時間が TCP に与える影響を排除し，省電力モードを用いないときと同等のスループットを得ることができる．省電力モードではモバイル端末が ACK を返すため基地局の送信キューに蓄積されるフレームの数は少ないのに対し，基地局が疑似 ACK を送信すると，モバイル端末のスリープ時間を待つことなく ACK が固定端末に届くため固定端末はモバイル端末のスリープ状況とは関係なくデータ送信を行うことになり，基地局の送信キューにはフレームが次々に蓄積されることになる．モバイル端末は TIM 間隔でスリープ状態から復帰する際に基地局に蓄積されたフレームをすべて受信するため，基地局に蓄積されるフレーム数が多くなるとスループットが改善する（図 5.4.1）．

・TCP セマンティクスの保護

フロー単位のセマンティクスを保護するためには，TCP 接続の維持と未 ACK セグメントの保護が必要になる．

TCP 接続を維持しフロー単位のセマンティクスを保つために，基地局は固定端末が FIN を送ってきてもこれに応答せず，モバイル端末が FIN 応答を発するまで TCP 接続を維持する．

未 ACK セグメントの保護とは，モバイル端末が ACK を返していないセグメントを基地

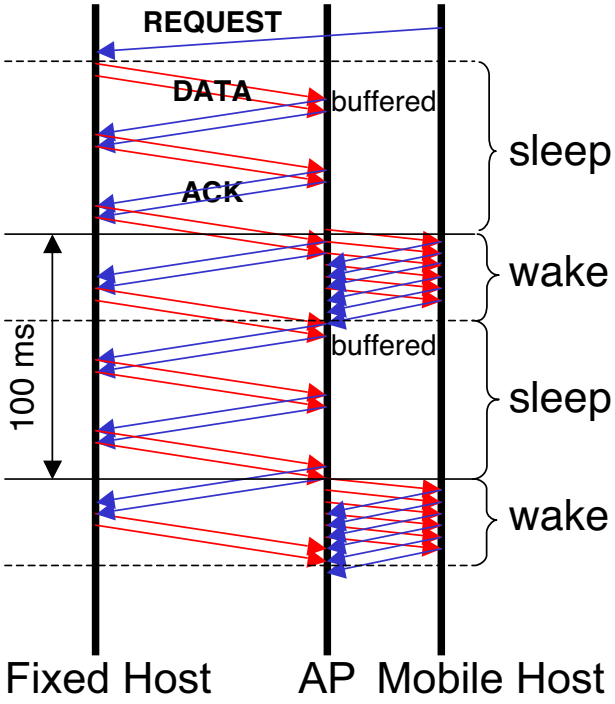


図 5.4.1 TCP スループット改善のアプローチ

局が確実にモバイル端末に届けることである．これは，疑似 ACK の生成によって固定端末がモバイル端末に対して該当セグメントの再送処理を停止するために必要となる．確実なデータ伝送を実現するためには，基地局からモバイル端末への伝送の際のパケットロスと，モバイル端末のハンドオーバ時のパケットロスに留意しなければならない．基地局とモバイル端末間でのパケットロスを防ぐために，基地局は固定端末から送られてきたすべてのパケットを保存し，モバイル端末が重複 ACK を送信してきたときに再送できるようにする．モバイル端末のハンドオーバ時には，モバイル端末が ACK を返していないデータが基地局に残っている可能性があるため，このデータを移動元基地局から移動先基地局に転送しハンドオーバ時のパケットロスを防ぐ．

5.4.2 省電力モバイル端末の TCP スループット改善手法

省電力モバイル端末の TCP スループット改善手法は，大きく三つの機構によって構成される．セマンティクスを保護するために TCP 接続を維持しながら無線区間でのパケットロスを保護し省電力モード時の高いスループットを確保する疑似 ACK 送信機構 (Pseudo-ACK Generator)，モバイル端末のハンドオーバ時に未 ACK セグメントを保護する基地局間連携機構（Data Forwarder)，そして疑似 ACK 送信機構と基地局間連携機構が動作するために必要な情報を管理する TCP 接続情報管理機構（TCP Connection Information Manager）である（図 5.4.2)．

以下では，簡単のためモバイル端末がハンドオーバしても IP アドレスは変化しないものとし，論を進める．

（1）TCP 接続情報管理機構

TCP 接続情報管理機構は，基地局を通過する TCP 接続に関する情報を管理し，疑似 ACK 送信によって固定端末から送信されてきた TCP パケットの複製を保存する機構である．

基地局を通過する TCP 接続に関する情報は，TCP 接続情報データベース（CIT：TCP Connection Information Table）で管理しており，このデータベースでは TCP 接続の IP アドレス，ポート番号，シーケンス番号，ウィンドウサイズ，TCP オプション，モバイル端末の MAC アドレス，保存してある TCP パケットのパケット数が管理されている．CIT には新しい TCP 接続が検出されるとエントリが追加され，該当する TCP 接続のパケットが基地局を通過するたびに更新される．TCP 接続の切断が検知された時とモバイル端末がハンドオーバしハンドオーバ先の基地局に TCP 接続情報を受け渡した時に，CIT から該当エントリが削除される．

一方，疑似 ACK 送信によって固定端末から送信されてきた TCP パケットを未 ACK セグメントの保護のために複製保存するのが TCP パケットストレージ（TCP Packet Storage）である．TCP パケットストレージは, 以下で述べる疑似 ACK 送信機構と連動し, TCP パケットが到着するとこれを TCP パケットストレージに複製保存し，保存している TCP パケットに対する ACK がモバイル端末から返ってくると TCP パケットストレージから該当パケッ

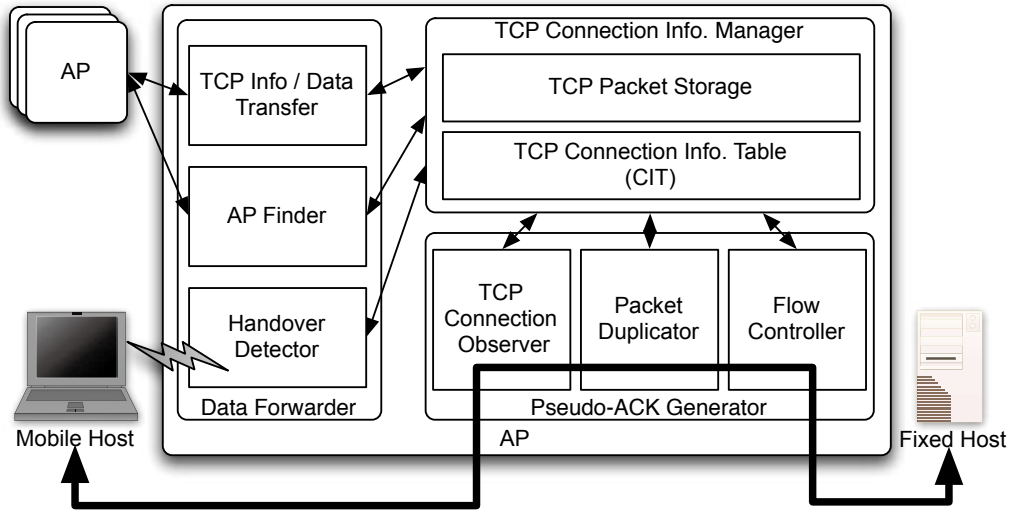

AP
TCP Info / Data Transfer
AP Finder
Handover Detector
Data Forwarder
TCP Connection Info. Manager
TCP Packet Storage
TCP Connection Info. Table (CIT)
TCP Connection Observer
Packet Duplicator
Flow Controller
Pseudo-ACK Generator
AP
Mobile Host
Fixed Host

図 5.4.2 構成図

トを削除する．またモバイル端末のハンドオーバ時には，基地局間連携機構と連携して保存している TCP パケットをハンドオーバ先の基地局に転送した後，TCP パケットストレージから削除する．

（2）疑似 ACK 送信機構

疑似 ACK 送信機構は三つのモジュールによって構成される．基地局を通過するフレームを監視し基地局を流れる TCP 接続状態を把握する接続状態監視モジュール（TCP Connection Observer），基地局を通過する TCP パケットをすべて複製するパケット複製モジュール（Packet Duplicator），および疑似 ACK を生成し送出するフロー制御モジュール（Flow Controller）である．

疑似 ACK 送信機構のパケット処理について，モバイル端末側から入力されたパケットの扱いと固定端末側から入力されたパケットの扱いに分けて，それぞれのフローチャートを図 5.4.3 と図 5.4.4 に示す．

・接続状態監視モジュール

接続状態監視モジュールは基地局を通過するフレームを監視し省電力モードを有効化したモバイル端末の TCP 接続を検出し処理を振り分ける．処理は，SYN パケットと FIN パケットに基づいた TCP 接続の開始と終了の処理，TCP 接続が開始後のデータパケットと ACK パケットの処理に大きく分けられる．

まず, TCP 接続の開始と終了の検出について述べる．TCP 接続の開始（図 5.4.3, 5.4.4(1)）は，SYN パケットから始まるスリーウエイハンドシェークによって検出し CIT に登録する．CIT には，TCP 接続の IP アドレス，ポート番号，シーケンス番号，ウィンドウサイズ，TCP オプション，モバイル端末の MAC アドレスを登録する．FIN パケットによる TCP 接続の終了要請が検出されると（図 5.4.3, 5.4.4(2)），CIT を参照し該当する TCP 接続の有無を確認し CIT に FIN フラグを立てる．すでに FIN フラグが有効になっている場合は FIN に対する ACK であると判断し，保存している TCP パケットがないことを確認して CIT から情報を削除する．

次に，TCP 接続開始後から終了までの処理について述べる．

モバイル端末側から入力されたパケットについて（図 5.4.3(3)）は，まず，TCP パケットストレージが該当パケットを削除できるようにパケット複製モジュールに ACK のシーケンス番号を通知する．重複 ACK の場合は基地局とモバイル端末の間でパケットロスが発生しているので, 未 ACK セグメントを保護するためにパケット複製モジュールから該当パケットを複製しモバイル端末に再送する．次に，受け取ったパケットの TCP ペイロードにデータが存在する場合はパケットに記載された ACK のシーケンス番号を更新して固定端末側に送信し, データが存在しない場合はパケットを破棄する．いずれの場合も, パケットからウィンドウサイズ情報を取り出し CIT を更新する．このウィンドウサイズ情報はフロー制御モジュールによって利用される．

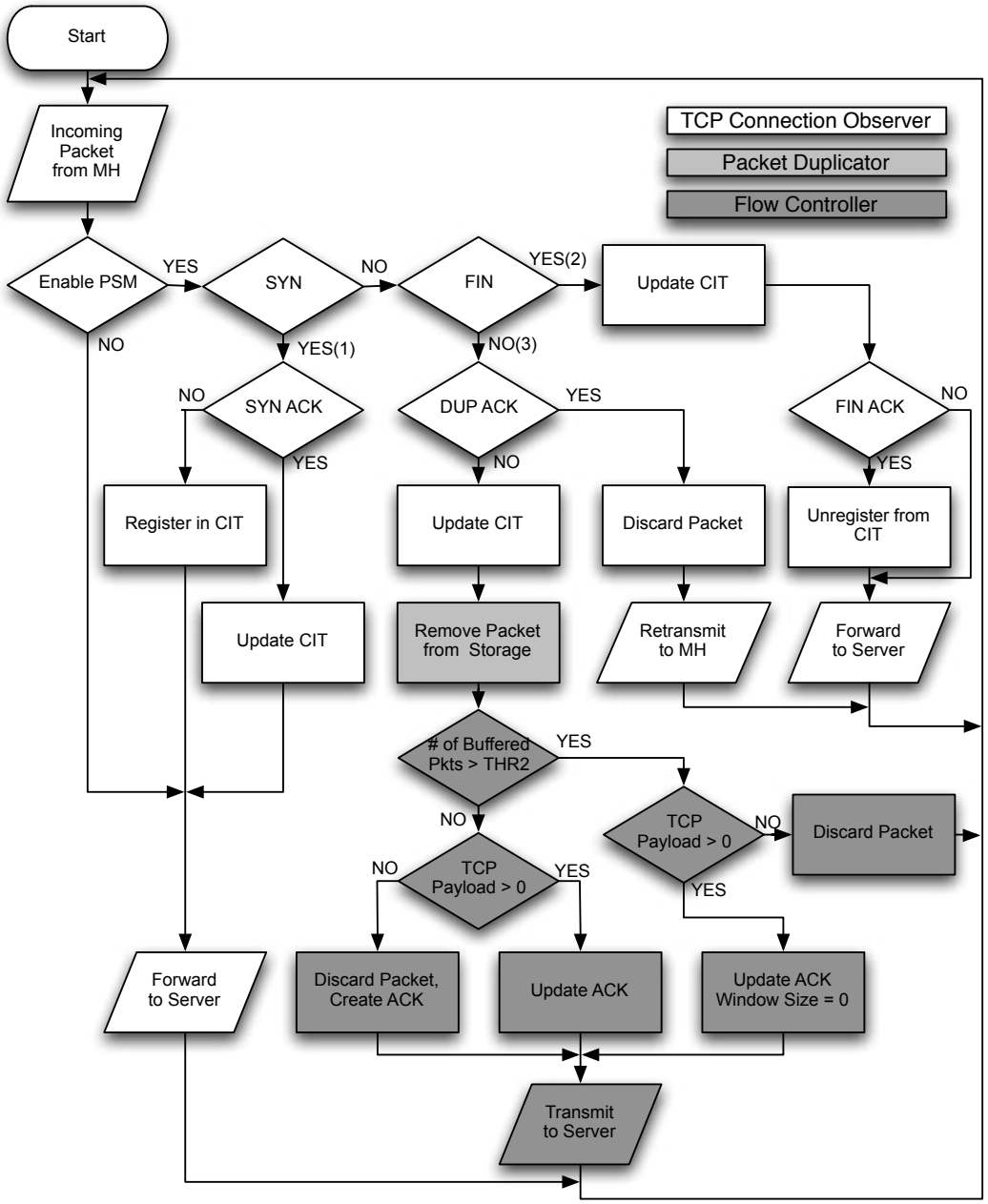


図 5.4.3 パケットの処理フロー（無線インタフェースからの入力）

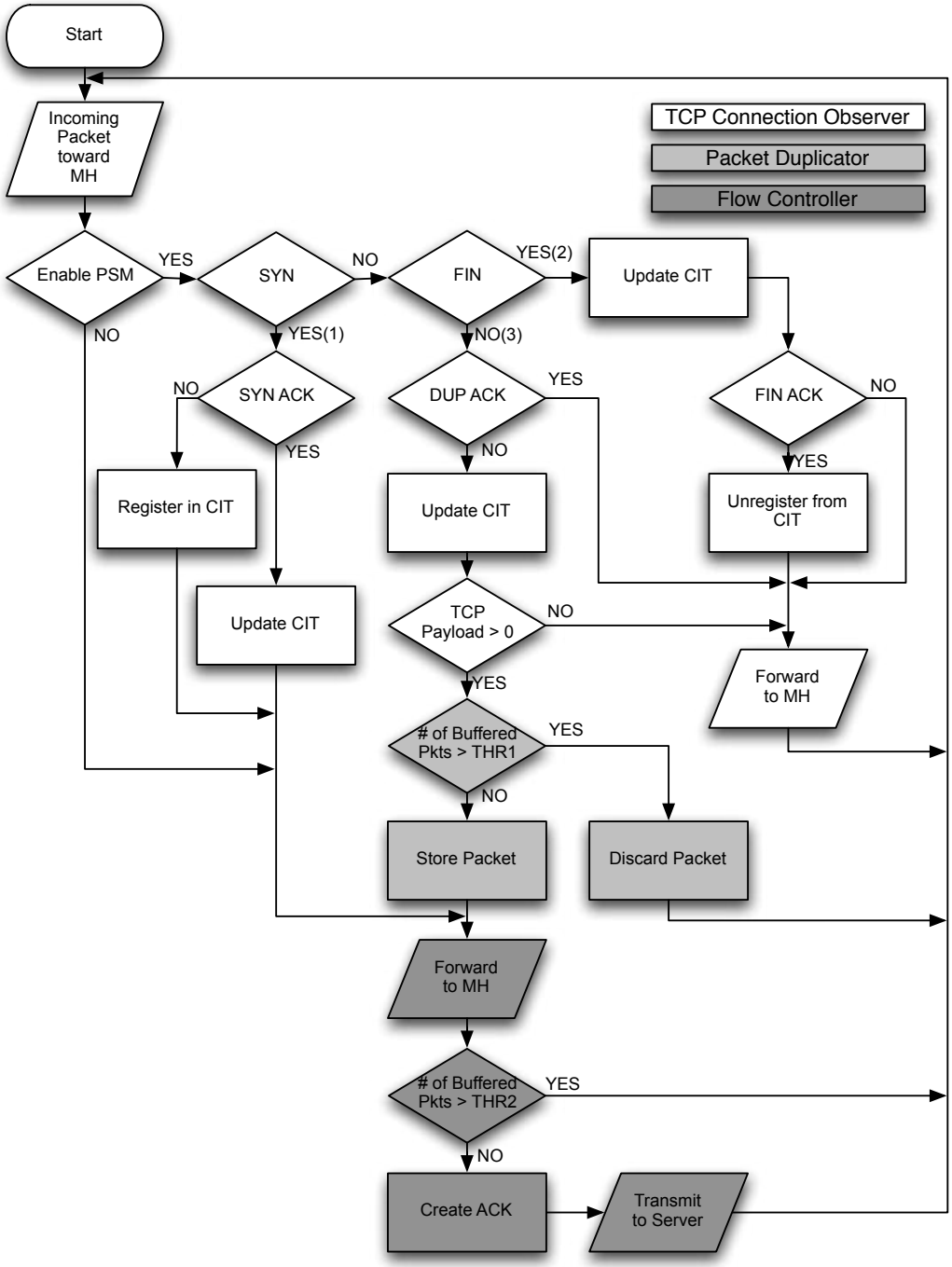


図 5.4.4 パケットの処理フロー（有線インタフェースからの入力）

固定端末側から入力されたパケットについて（図 5.4.4(3)）は，重複 ACK でなければ CIT の情報を更新して，TCP ペイロードにデータがありなおかつ TCP パケットストレージに空き容量があればパケットを複製しパケットをパケット複製モジュールに渡すとともにモバイル端末に転送しフロー制御モジュールを動作させる．重複 ACK の時はモバイル端末に転送する．TCP パケットストレージに空き容量がない場合はパケットを破棄する．TCP ペイロードにデータがない ACK パケットは複製せずにモバイル端末に転送する．TCP パケットストレージの空き容量はパケット数で管理する．

・パケット複製モジュール

パケット複製モジュールは，CIT に登録された TCP 接続の複製 TCP パケットを保存管理する．接続状態監視モジュールから複製パケットを受け取ると，TCP パケットストレージに保存し CIT の総パケット数を更新する．接続状態監視モジュールが ACK パケットのシーケンス番号を通知してきたときに重複 ACK を検出すると，該当する TCP パケットを接続状態監視モジュールに受け渡す．重複 ACK でない場合は TCP パケットストレージから該当するパケットが削除され，CIT の総パケット数が更新される．なお，総パケット数が TCP パケットストレージの最大容量（THR1）に達している場合，到着パケットは破棄される．

・フロー制御モジュール

フロー制御モジュールはパケット複製モジュールが書き込んだ CIT のパケット数を参照しながら疑似 ACK を生成して TCP の流量制御を行い，基地局に保存しているパケット数をフィードバック制御する．フィードバック制御は疑似 ACK 生成閾値（THR2）に基づいて行う．すなわち，TCP パケットストレージに保存されているパケット数が疑似 ACK 生成閾値に達していない場合，疑似 ACK を生成し次の TCP パケットの送信を固定端末に要請する．疑似 ACK に記載する受信ウィンドウサイズにモバイル端末の受信ウィンドウサイズを反映させるために CIT の受信ウィンドウサイズ情報を利用する．保存されている TCP パケットが疑似 ACK 生成閾値を超えている場合は，CIT に記載されているモバイル端末の受信ウィンドウサイズ情報とは無関係に ACK に記載する受信ウィンドウサイズを 0 にすることで TCP の送信レートに影響を与えることなく固定端末からの送信を一時的に抑制し，保存する TCP パケット数を制御する．

（3）基地局間連携機構

基地局間連携機構はモバイル端末のハンドオーバ時にのみ機能する機構である．ハンドオーバ時の処理シーケンスを図 5.4.5 に示す．まずモバイル端末が基地局にアソシエーションしたのを検知し（1），周辺基地局にモバイル端末の移動元基地局を問い合わせる．次に，移動元基地局に対して TCP 接続情報と保存している TCP パケットの転送を依頼し（2），移動元基地局からこれらの情報を受け取る（3 - 4，6）．受け取った情報に基づいて TCP 接続情報を復元し（5），保存してある TCP パケットをモバイル端末に向けて送信するとと

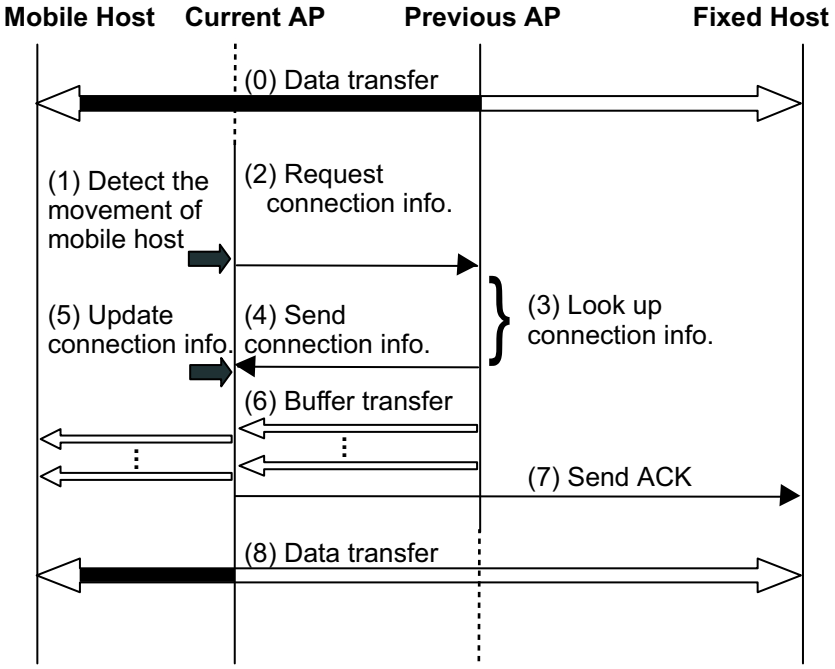


図 5.4.5 ハンドオフ時のシーケンス

もに，疑似 ACK 送信機構のフロー制御モジュールを呼び出して，疑似 ACK を生成し固定端末に対して送信することで固定端末に対してデータ送信を促す（7）.

基地局間連携機構は三つのモジュールから構成される．データリンク層を監視し，新たにアソシエーションを構築したモバイル端末を速やかに検知する移動検知モジュール (Handover Detector)，アソシエーション情報もしくはモバイル端末の MAC アドレスを用いて移動元の基地局を発見する基地局発見モジュール（AP Finder)，該当する端末の TCP 接続に関する情報と保存されている TCP パケットを転送する転送モジュール（TCP Info/ Data Transfer）である.

・移動検知モジュール

移動検知モジュールはモバイル端末の移動をデータリンク層で検知する機構である．モバイル端末が送信するアソシエーションメッセージとリアソシエーションメッセージを監視し，基地局に繋がっているモバイル端末を管理し，新しい端末が接続されたときには，新しい端末の MAC アドレスを基地局発見モジュールに通知する．リアソシエーションメッセージにより移動元の基地局の MAC アドレスが分かる場合には，端末の MAC アドレスとあわせて基地局発見モジュールに移動元基地局の MAC アドレスを通知する.

・基地局発見モジュール

基地局発見モジュールは近隣基地局の情報を保持する．具体的には，近隣基地局の MAC アドレスと IP アドレスのテーブルを管理する．移動元基地局の MAC アドレスがわかる場合は，テーブルを参照して移動元基地局の IP アドレスを求める．また，移動元基地局の MAC アドレスが不明な場合は，移動検知モジュールから通知を受けて，新しく接続した端末の MAC アドレスを元に，近隣の基地局に問い合わせを行い移動元基地局を検索し，転送モジュールに端末の MAC アドレスと移動元基地局の IP アドレスを通知する.

・転送モジュール

転送モジュールは基地局発見モジュールから新たに基地局とアソシエーションを構築したモバイル端末の MAC アドレスと移動元基地局の通知を受け，移動元基地局に TCP 接続情報と保存している TCP パケットの転送を要求する．転送された TCP 接続情報は自基地局の CIT に書き込み, TCP パケットは TCP パケットストレージに保存する．また, 転送モジュールは他の基地局からの転送要求も受け付ける．転送要求を受けた場合には，該当する MAC アドレスの TCP 接続状態と保存している TCP パケットを要求元基地局に転送する．転送完了後，CIT の TCP 接続情報と TCP パケットストレージに保存された該当する TCP パケットの削除をパケット複製モジュールに要請する.

5.5 実装および評価

5.5.1 モビリティ対応省電力 TCP プロキシの実装

筆者は，上述の省電力モバイル端末のための TCP スループット改善手法を実装し性能評価を行った．実装システムを図 5.5.1 に示す．本実装はプロトタイプ実装のため，実際の基地局上ではなく基地局に直接接続したデスクトップ PC を用いて動作検証をおこなった．このため，モバイル端末のアソシエーションを検知する移動検知モジュールの実装は，モバイル端末から移動検知モジュールに対してメッセージを送ることとした．

本実装では，接続状態監視モジュールが必要とするパケット取得には divert ソケットを用い，取得されたパケットはすべてアプリケーションプロセスで処理した．パケット複製モジュールの TCP パケットストレージの容量は TCP 接続毎に 50 パケット分用意し，基地局が疑似 ACK を作成する疑似 ACK 生成閾値を 30 パケットとした．疑似 ACK の作成には raw ソケットを利用した．

本実装に用いた機器の仕様は次の通りである．デスクトップ PC は，CPU が Pentium III 800MHz で，OS は FreeBSD 4.7 を用いた．モバイル端末の CPU は Pentium II 233MHz で，OS は Plamo Linux2.2.5 である．固定端末の CPU は Pentium II 233MHz で，OS は Debian GNU/Linux 3.0 である．基地局には，AVAYA の IEEE 802.11b 基地局 Access Point III を用いた．基地局と固定端末の間には, FreeeBSD 4.9 で構築した DUMMYNET [8] を用いた通信遅延装置を設置した．

5.5.2 評価

疑似 ACK 送信時の TCP スループットとハンドオーバ時の基地局間連携動作について動作検証と評価を行った．

まず疑似 ACK 送信による TCP スループット改善を確認するために，モバイル端末のハンドオーバを想定せずに TCP スループットを評価した（表 5.5）．TCP スループットの評価はファイル転送によって行った．ファイルサイズは 50KB と 500KB の二種類とし，通信遅延装置に設定する片道通信遅延を 10 ミリ秒，20 ミリ秒，40 ミリ秒と変化させながら測定を行った．表には，左から順に省電力モードを無効にした場合（Non-PSM），省電力モードを有効にした場合（PSM），省電力モードで本手法を利用した場合（Our method）の TCP スループットをまとめた．表から，本手法は省電力モードを有効にしたときよりもスループットが高く，省電力モードを無効にした場合と比較してもスループットの低下は 0\% から 15\% 程度であり，本手法の有効性が確認された．

次にハンドオーバ時の動作検証を行った．動作検証として，モバイル端末がハンドオーバ後でも TCP 接続を継続して利用できていることを確認した．

最後にハンドオーバ時の基地局間連携によって発生するハンドオーバ遅延時間について述べる．ハンドオーバ時の総遅延時間（図 5.4.5 の (1)-(8) に対応）は平均 1.66 秒であった．

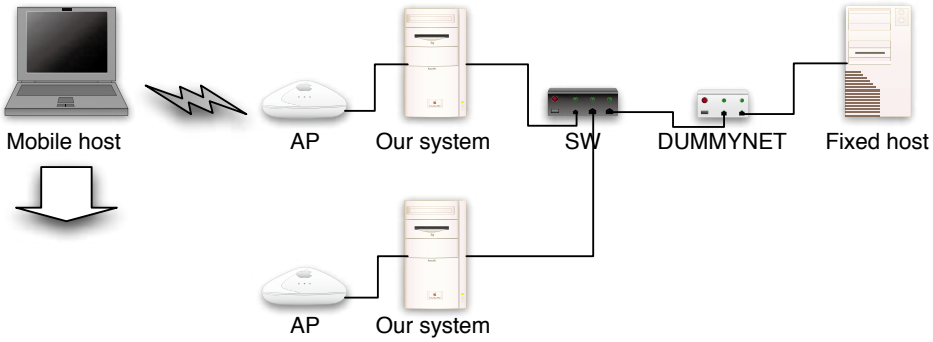


図 5.5.1 実装構成図

表 5.5 TCP スループット性能（上段：50KB 転送，下段：500KB 転送）[Mbps]

One-way delay	Non-PSM	PSM	Our method
10 ms	1.51	0.75	1.30
	3.20	1.67	3.02
20 ms	0.88	0.56	0.86
	1.82	1.39	1.78
40 ms	0.46	0.38	0.46
	0.97	0.76	0.86

遅延時間の内訳としては，TCP 接続情報の転送処理遅延（図 5.4.5 の (1)-(5) に対応）は，平均 4.7 ミリ秒であり，TCP パケットの転送処理を含めた遅延（図 5.4.5 の (1)-(7) に対応）は 10 ミリ秒未満であった．このとき基地局間で転送された TCP パケット数は平均 41 パケット，60KB である．最も大きい遅延要素は，基地局間で TCP パケットを転送した後に固定端末からデータが送られてくるまでの時間であった．この遅延時間は，ハンドオフ時に固定端末側で TCP の再送タイムアウトが発生しているためである．これは，固定端末はモバイル端末にむけてパケットを送信するものの，ハンドオーバ時にはモバイル端末，移動元基地局ともに TCP セマンティクスを維持するためにパケットを受け取らず ACK も送信しないことが原因である．

5.6 むすび

本章では，セッション層が管理する通信情報の範囲を明らかにするため，トランスポート層プロトコルの内部状態を管理する IEEE 802.11 省電力モバイル端末における TCP スループットの改善手法について述べた．

省電力モードでは，モバイル端末がスリープ状態の間，基地局がフレームを一時的に蓄積するため，TCP の通信遅延時間の増加を招きスループットが劣化する．これまでモバイル端末に機能追加することで TCP スループットの改善を図る研究が提案されてきたが，筆者は基地局に疑似 ACK 送信機能と基地局間連携機能を搭載することでスループットの改善を図った．疑似 ACK 送信機能により固定端末にデータ送信を促すことでスループットの改善を実現し，基地局間連携機能によりモバイル端末がハンドオーバした際の TCP のセマンティクスの維持を可能にしている．最後に，手法の実装と評価について述べ，省電力モードで動作するモバイル端末の TCP スループットが改善しハンドオーバ時でも動作することを示した．

セッション層がトランスポート層プロトコルの内部状態を管理する場合，5.4，5.5 で示したようにシステムは複雑になる．さらに，様々なトランスポート層プロトコルごとに内部状態を管理すると，システムはさらに複雑になる．すなわち，本章は，セッション層がトランスポート層プロトコルの内部状態を管理するべきではないことを示唆している．

参考文献

[1] Institute of Electronic and Electrical Engineers (IEEE). "Wireless medium access control (MAC) and physical layer (PHY) specifications." Standard 802.11, 1999.

[2] B. McFarland and M. Wong, "The Family Dynamic of 802.11," ACM Queue, Vol.1, No.3, pp.28--38, 2003.

[3] livedoor wireless, "http://wireless.livedoor.com/,".

[4] Lucent techonologies, "IEEE 802.11 WaveLAN PC Card user's guide," pp.A-1.

[5] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," In Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS), pp.136--143, 1995.

[6] H. Balakrishnan, S. Seshan, E, Amir, and R. Katz, "Improving TCP/IP performance over wireless networks," In Proceedings of the 1st International Conference on Mobile Computing and Networking (MOBICOM), 1995.

[7] O. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access using bounded slowdown," In Proceedings of the 8th International Conference on Mobile Computing and Networking (MOBICOM), pp.119-130, September 2002.

[8] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," ACM Computer Communication Review, Vol.27, No.1, pp.31--41, 1997.

6 セッション層の遠隔制御

6.1 はじめに

コンピュータの低廉化と小型化に伴って，ユーザが複数のコンピュータを利用することは珍しくない．今後，この傾向はさらに進み，ユーザが同時に複数のコンピュータを利用することが日常的になるであろう．コンピュータの同時複数利用は，異なるコンピュータでアプリケーションプログラムが同時複数利用されることを意味しており，ユーザ主導型通信の有効性が発揮される．

本章では，複数のコンピュータの同時利用を実現するためのセッション層アーキテクチャについて述べる．セッション層アーキテクチャは，３で述べたセッション層を遠隔制御することができるアーキテクチャである．セッション層の遠隔制御とは，具体的にはそれぞれのコンピュータ上で動作するセッション層制御部の機能をアプリケーションプログラムが動作するコンピュータから独立させ，さらにユーザごとに一つに集約することである．

まず 6.2 でコンピュータの同時複数利用について述べる．つぎに，6.3 でセッション層の遠隔制御における課題について述べ，6.4 でセッション層の遠隔制御を実現するセッション層アーキテクチャについて述べる．そして 6.5 でセッション層アーキテクチャの実装と評価について述べ，6.5 で関連研究について述べる．

6.2 コンピュータの同時複数利用

近年，コンピュータの利用形態が，１人のユーザがひとつのコンピュータを使うモデルから１人のユーザが複数のコンピュータを利用するモデルに大きく変化しつつある．このようなコンピュータの利用形態の変化は，通信サービスにも大きな影響を与えることになる．具体的には，一つの通信サービスを複数の通信デバイスで実現したり，複数の通信を同時に並行して行うことになるであろう．例えば，テレビ電話を複数のディスプレイやカメラ，マイク，スピーカで実現したり，自宅やオフィスの状況を常に監視するといったサービスである．

これまでに開発された複数の通信デバイスを用いたより簡単な通信の実現方法は，複数の通信アプリケーションを統合的に管理するアプリケーションシステムである．例えば，複数のマルチメディア通信デバイスを用いて遠隔会議を行える Access Grid [1] は，Access Grid Tool Kit を用いてアプリケーションプログラムを作成し制御システムに登録することで，制御システムがアプリケーションプログラムを制御することが可能になり，複数の通信デバイスを統一的に利用することができる．

しかし，一人で複数の通信デバイスを利用する通信は，通信デバイスの低廉化や小型化が進んでいるにも関わらず，あまり現実のものになっていない．これは，現在の通信モデルでは，ユーザがそれぞれの通信デバイスで動作するアプリケーションプログラムに通信の相手先情報を直接入力しなければならないため，複数の通信デバイスを用いるには手間がかかるからである．上述の Access Grid では通信の相手先情報をアプリケーションプログラムに自動転送することができるが，利用するためにシステムとアプリケーションの間で事前の認証やセットアップが必要であり，場所や時間を問わずに利用できるものではない．

すなわち，アプリケーションプログラムを通して通信の制御を行う手法は，複数の通信を様々な通信デバイス上で同時に動作させるのには向いていない．

３で述べたセッション層の技術では，アプリケーションプログラムから通信の制御を切り離してユーザが直接通信を制御することが可能であり，ユーザがアプリケーションプログラムに自由に接続，切断することができる．次節以降では，複数のコンピュータ上のセッション層を遠隔から集中的に管理する機構について述べる．

6.3 セッション層の遠隔制御における課題

まず，セッション層を単一の通信デバイスで用いる場合をセッション層の局所制御モデルとして示す（図 6.3.1）．局所制御モデルでは，セッション層 API，セッション層ミドルウエア, セッション層制御部の三つのコンポーネントが単一の通信デバイス上で動作する. セッション層 API は, アプリケーションプログラムが利用する通信インタフェースの API であり, API を通してセッション層ミドルウエアと通信を行う．セッション層ミドルウエアは，セッション層制御部からの指示にしたがって，TCP や UDP を用いて通信チャネルの構築，切断を行う．セッション層制御部は，通信相手のセッション層制御部と認証を行い，通信チャネルの構築に必要な情報を交換する．

次に，セッション層の遠隔制御モデルの基本構成について述べる（図 6.3.2）．セッション層の遠隔制御では，セッション層制御部をセッション層ミドルウエアが動作する通信デバイスから切り離し，独立した通信デバイス上で動作させる．セッション層制御部はユーザ毎にひとつとし，これが複数の通信デバイスで動作するセッション層ミドルウエアに指示を送る．なお，ユーザはセッション層制御部を通して制御を行うことは遠隔制御モデルでも同様である．

すなわち，セッション層の局所制御モデルと遠隔制御モデルの違いは，セッション層制御部がセッション層ミドルウエアと異なる通信デバイス上で動作することである．この相違により，以下の三つの技術課題が発生する．

・利用可能アプリケーションプログラムの把握

それぞれの通信デバイスで様々なアプリケーションプログラムが動作している．したがって，ユーザはどの通信デバイスでどのようなアプリケーションプログラムが動作しているかを把握する必要がある．局所制御モデルではセッション層制御部が直接セッション層ミドルウエアにアクセスし動作しているアプリケーションプログラム情報を入手することができるが，遠隔制御モデルではセッション層制御部はどの通信デバイスのセッション層ミドルウエアにアクセスするのかすら決まっていない．さらに，遠隔制御モデルでは，ユーザが通信デバイスにアクセスが可能で，実際に利用権限をもっているアプリケーションプログラムかどうかということも判断する必要がある．

・通信チャネル状態の把握

ユーザが利用するアプリケーションサービスが用いる通信チャネルは，アプリケーションプログラムが動作する通信デバイスで動作するセッション層ミドルウエアが終端している．セッション層制御部はセッション層ミドルウエアとは別の通信デバイスで動作しているため，セッション層ミドルウエアが終端している通信チャネルが利用可能であるかを把握する必要がある．

例えば，TCP を用いて通信チャネルを構築していた場合，通信デバイスのインターネッ

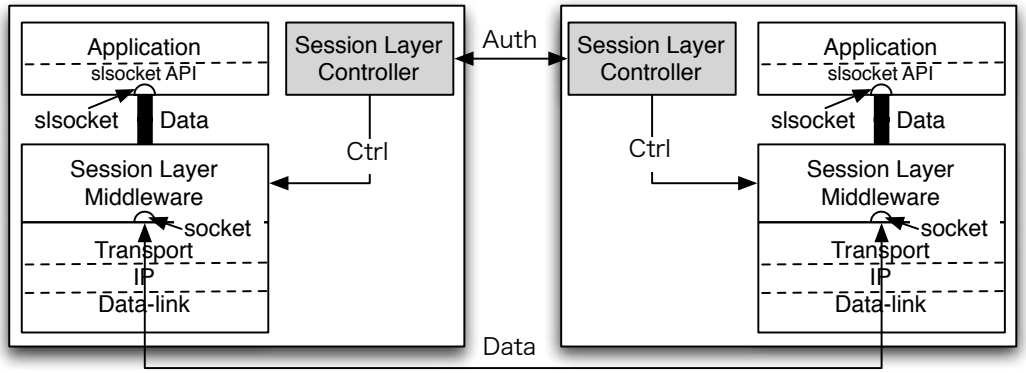


図 6.3.1 セッション層の局所制御モデル

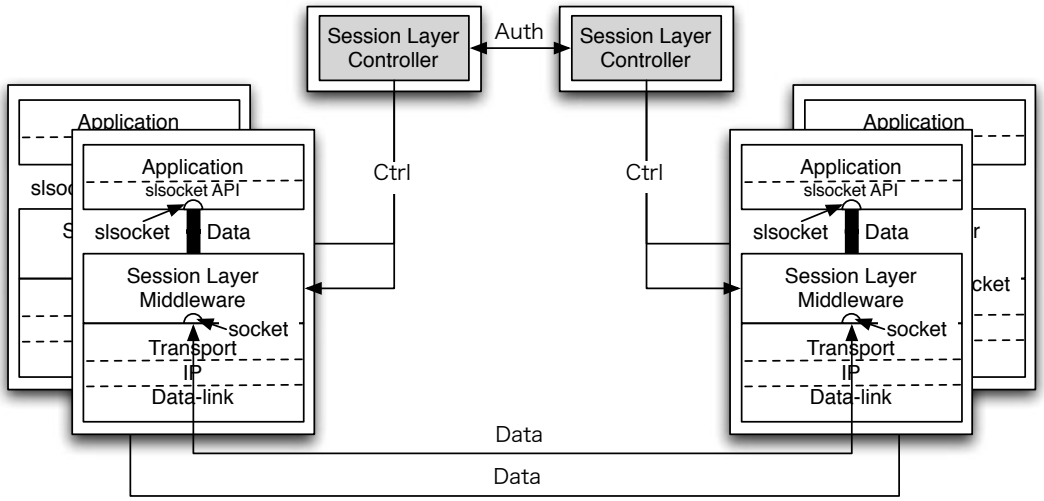


図 6.3.2 セッション層の遠隔制御モデル

ト接続環境が劣悪でセッション層制御部が切断の指示を出していないのに TCP 接続が切断されることがある．また，UDP はコネクションを保持しないトランスポート層プロトコルのため，セッション層ミドルウエアはセッション層制御部から切断指示が発行されない限り通信インタフェースを保持しアプリケーションプログラムからのデータをネットワークに送信し続ける．しかし，通信デバイスがネットワークから切り離された場合や通信相手が通信を終了したときには，構築した通信チャネルが利用可能な状態ではないことを把握する必要がある．

・制御用通信の確保

局所制御モデルではセッション層制御部とセッション層ミドルウエアが同一の通信デバイスで動作しているため，セッション層制御部とセッション層ミドルウエア間の通信は信頼性があり，またいつでも制御できる．しかし，遠隔制御モデルではセッション層制御部とセッション層ミドルウエアは異なる通信デバイス上で動作するため，一般にこれらの間の通信は信頼性がなく，いつでも制御できる状態にあるとはいえない．したがって，セッション層制御部とセッション層ミドルウエアとの間の制御用通信を確保する必要がある．

6.4 セッション層アーキテクチャ

セッション層は，多様な通信デバイスが遍在する環境において，ユーザが簡単かつ安全に通信を行うためのサービスフレームワークであり，通信に使用するデバイスやネットワークへの接続インタフェースに依存しない柔軟な通信サービス利用を可能にする．具体的には，ユーザが通信サービス利用において直接使用する機器とは異なる通信制御専用の機器を導入する．この通信制御専用の機器はユーザの通信をデバイスやアプリケーションにかかわらず一括して管理することで，ユーザの利用する多様な通信デバイス間の差異を吸収し，通信の開始や終了および通信内容の変更を統一的なインターフェースの元でユーザに提供する．

まず，図 6.4.1 にセッション層アーキテクチャの全体構成を示す．図から明らかなようにセッション層アーキテクチャは三つの要素から構成される．

・実通信端末

ユーザが実際のネットワークサービス利用に使用する端末を，実通信端末と呼ぶ．なお，ユーザの通信サービス利用において実通信端末間の通信を実通信と呼ぶ．実通信端末ではセッション層ミドルウェアが動作し，このミドルウェアとアプリケーションプログラム間の通信を実現する slsocket API によってアプリケーションプログラムから通信の下位情報が分離される．これによってアプリケーションにおける通信がデバイスやネットワークに非依存となり，アプリケーション開発が容易になるとともに，アプリケーションにおけるセキュリティが向上する．

・通信制御サーバ

ユーザの通信を一括して管理するのが通信制御サーバである．通信制御サーバはセッション層におけるセッション層制御部の機能を担い，ユーザの行う通信に関するネゴシエーションを通信制御サーバ同士で行うとともに，実通信端末上のセッション層ミドルウェアと連携し，ユーザの通信を管理する．セッション層ミドルウェアではアプリケーションと通信を明確に切り離しているため，通信制御サーバにおけるユーザの通信管理もアプリケーションに依存しない統一的なものである．通信制御サーバはユーザ自身が設置することも可能であるが，携帯キャリアやインターネットサービスプロバイダがユーザに対するサービスの一環として提供することを想定している．

・通信制御端末

通信制御端末は，サーバである通信制御サーバのユーザインターフェースとしての機能と，実通信端末の行うサービス広告を受信するサービス発見の機能を持つ．ユーザは通信制御端末を常に持ち歩くことを前提にしており，携帯電話等の小型端末に通信制御端末としての機能が載ることを想定している．

通信制御端末は通信制御サーバと併せてユーザの通信を管理，制御する機能を果たしてい

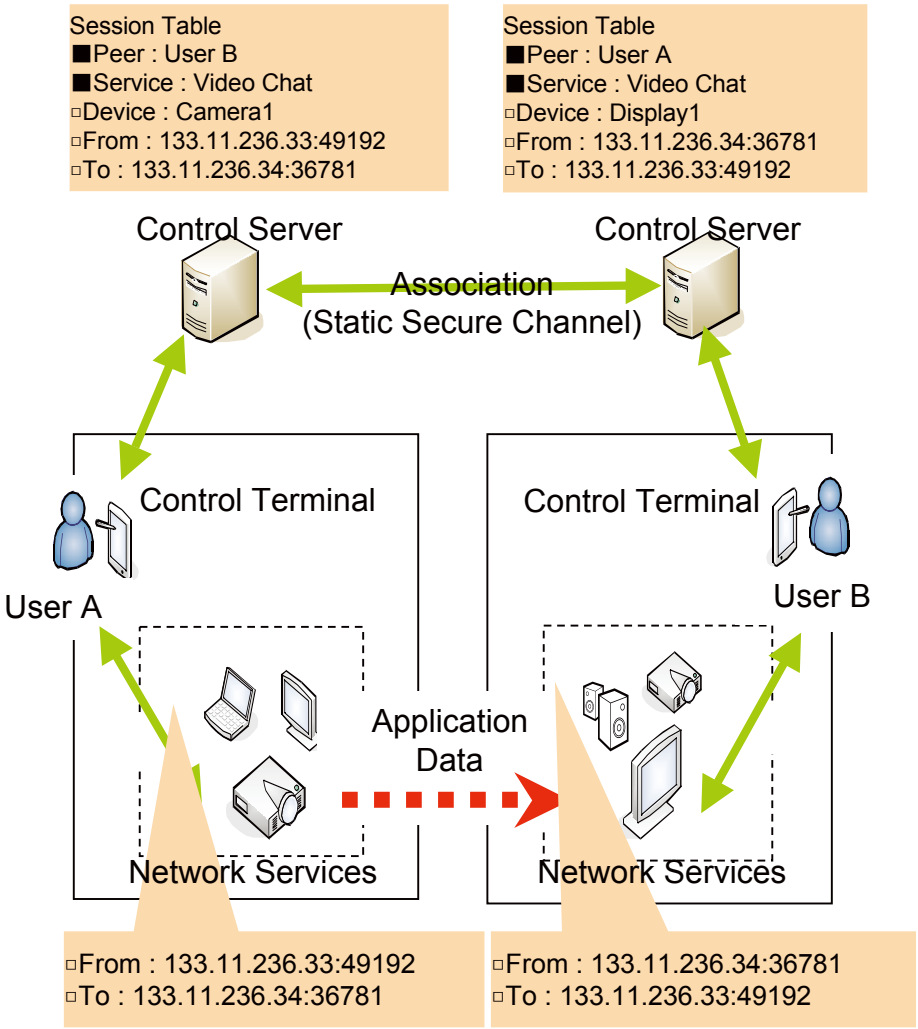


図 6.4.1 セッション層アーキテクチャ

るが，ここで通信制御端末と通信制御サーバを分離させている理由は二つある．一つは小型端末である通信制御端末では消費電力や容積の問題から計算資源が限られ，通信制御サーバとしての役割を果たすことが困難であるためである．もう一つの理由は，据え置き型のサーバと異なり電波を使った通信が中心となる小型端末では，ユーザの移動によって通信環境が大きく変化し，端末へのネットワーク到達性を常時確保することが難しいためである．

次に，前節で述べた技術課題をセッション層アーキテクチャではどのように解決しているのかについて述べる．

・利用可能アプリケーションプログラムの把握

実通信端末上で動作するアプリケーションプログラムは大きく二つに分類できる．一つはユーザがよく使うアプリケーションプログラムであり，もう一つは，ユーザが外出先などで高々数回利用するアプリケーションプログラムである．

よく使うアプリケーションプログラムは，ホームネットワークやオフィスネットワークなどユーザが良くアクセスする場所に存在すると考えられる．したがって，これらの情報はユーザが事前に入手し，通信制御端末や通信制御サーバにブックマークのような形で保存しておくことができる．

滅多に使わないアプリケーションプログラムは，外出先などで，新しくアプリケーションサービスを発見し利用するものであろう．この場合は，通信制御端末を使って，直接当該アプリケーションプログラムが動作する実通信端末から利用可能アプリケーションプログラム情報を入手することができる．

通信制御端末と実通信端末間の通信方法として IrDA や Bluetooth，RFID，無線 LAN など様々な方法が想定されるが，これらを限定する必要はない．また，アプリケーションプログラム情報を直接実通信端末から入手するのではなく，実通信端末を管理するサーバからネットワークを介してこれらのアプリケーションプログラム情報を入手する場合もあろう．

・通信チャネル状態の把握

通信チャネル状態を遠隔から把握する方法には三つの方法がある．

一つは，通信制御サーバと実通信端末の間に制御用の通信チャネルを常時準備しておき，実通信端末が通信チャネル状態を常に通信制御サーバに通知する方法である．この方法の特徴は，ほぼリアルタイムでチャネル状態を把握することができるが，通信制御サーバと実通信端末間の通信のオーバヘッドが非常に高い．

二つめの方法は，通信制御サーバが実通信端末に対してポーリングを行い定期的に通信チャネル状態を確認する方法である．この方法は，通信チャネル状態の把握の正確さがポーリング間隔によって決定するが，通信制御サーバと実通信端末間のオーバヘッドを軽減することができる．

三つめの方法は，通信チャネル構築時に通信制御サーバが設定した状況が発生したときに，

実通信端末が通信制御サーバに通知する方法である．この方法の特徴は，通信チャネルの細かな状態の把握はできないが，通信制御サーバと実通信端末間の通信オーバヘッドを最小限に抑えることができる．

これらは，通信制御サーバと実通信端末間のネットワーク帯域のみならず，ネットワークの安定性も考慮しアプリケーションプログラムに応じて設定されることが望まれる．

・制御用通信の確保

セッション層アーキテクチャでは，実通信端末と通信制御サーバ間の制御用通信として，直接通信と通信制御端末を介した間接通信の両方を併用する．

実通信端末と通信制御サーバが直接通信できるためには，これらの間でパケットを自由にやりとりできる必要がある．そのため，昨今のファイアウォールの普及を考慮すると，通信制御サーバと同じネットワークで動作する実通信端末に限られるであろう．

通信制御端末と通信制御サーバ間にセルラーネットワーク等の別ネットワークがあった場合，実通信端末と通信制御サーバの通信を通信制御端末が中継することで，実通信端末と通信制御サーバの間にファイアウォールがあっても実通信端末を制御することができる．

6.5 実装および評価

6.5.1 実装

3 で述べたセッション層の実装を拡張してセッション層アーキテクチャを実装した．拡張箇所は，通信制御サーバと通信制御端末の導入によるセッション層アーキテクチャの構築，およびセッション層アーキテクチャを用いたサービスモビリティサポートである．

まず，セッション層アーキテクチャの実装について述べる．通信制御サーバと通信制御端末の導入を新たに導入しセッション層制御部を通信制御サーバで動作させた．さらに，通信制御サーバと通信制御端末の間と通信制御端末と実通信端末の間の通信が切断されても再接続後に処理を継続できるようなラッパープログラムを動作させた．ラッパープログラムの導入により，通信制御端末が制御を行うたびに接続・切断処理を行っても処理が継続される．

通信制御端末で動作するプログラムは，ncell デーモンプログラムと cellular プログラムである（図 6.5.1）．cellular プログラムはユーザインタフェース部であり，ncell デーモンプログラムが通信制御サーバや実通信端末と通信を行っている．ncell デーモンプログラムは，通信制御サーバの制御ユーザインタフェースの信号と，通信制御サーバが実通信端末に制御する制御信号の二つの信号が流れるが，これらは別々に処理している．

次に，サービスモビリティを実現するために，通信制御サーバで動作するセッション層制御部が通信相手のセッション層制御部とやりとりする制御メッセージを拡張した．なお，実通信端末で動作するセッション層ミドルウエアの実装は，4 で述べたとおりである．

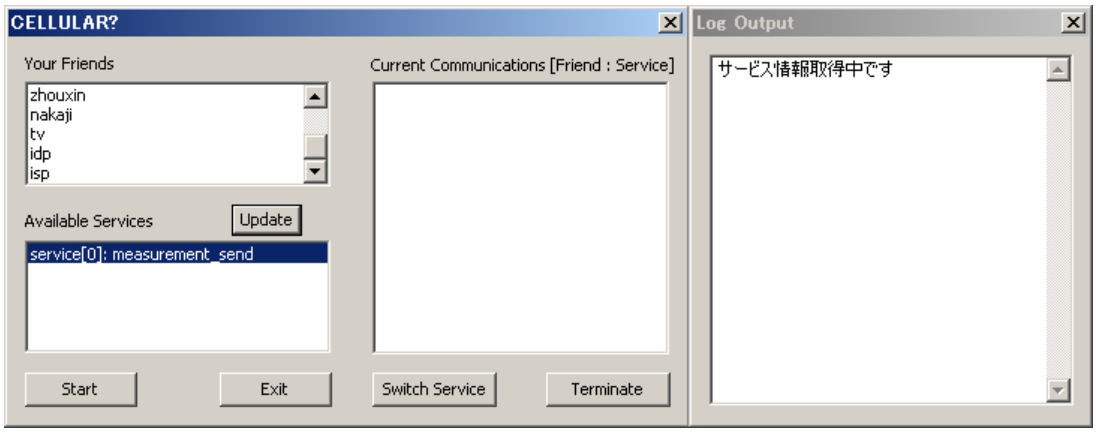
実装に用いた機器は以下の通りである．まず．通信制御サーバには，Intel Xeon 2.4GHz のプロセッサを二つ搭載した PC に FreeBSD 5.4 をインストールして用いた．通信制御端末には，Pentium M 1.1GHz のプロセッサを搭載した小型 PC に Windows XP をインストールして用いた．実通信端末は，Intel Xeon 2.4GHz のプロセッサを二つ搭載した PC に NetBSD2.1 をインストールして用いた．

ネットワーク構成として通信制御端末には，実通信端末と異なるネットワークを無線 LAN を用いて提供した．また，通信制御端末と実通信端末間のやりとりには IrDA を用いた．

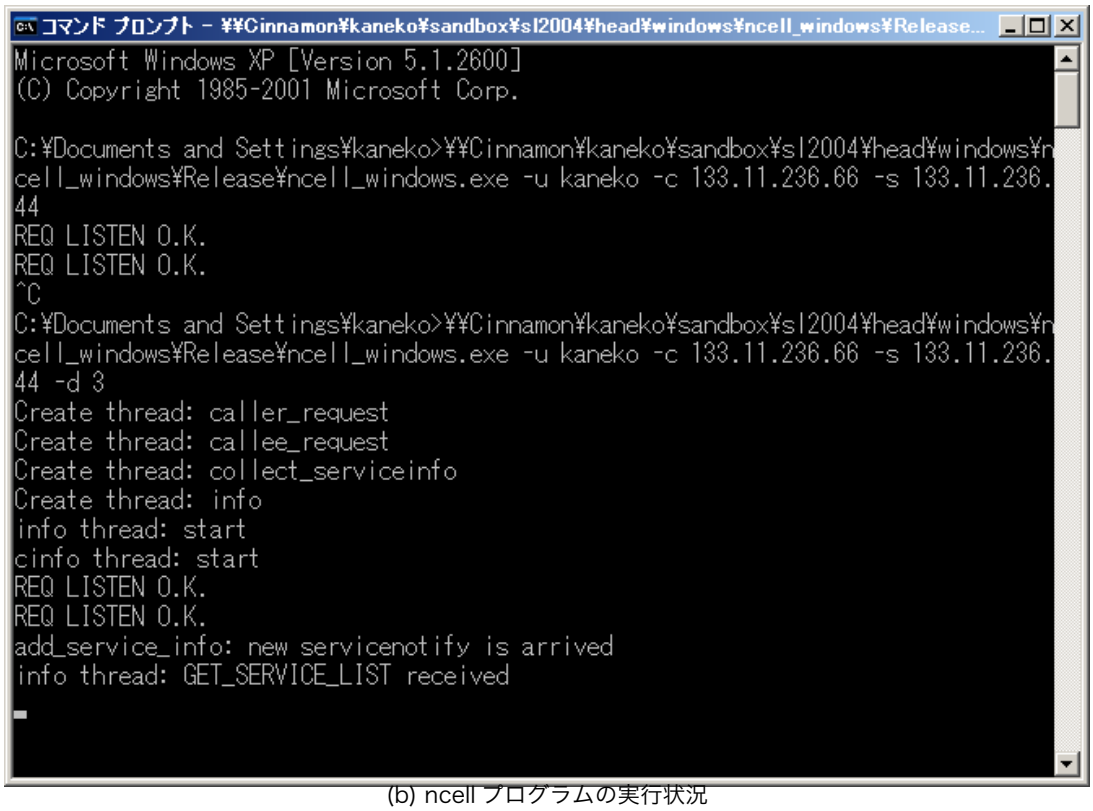
6.5.2 評価

まず，通信制御端末を用いた通信制御の動作確認を行った．通信制御端末と実通信端末間の制御情報のやりとりに無線 LAN を用いた場合，通信制御端末から通信の開始命令から実際に通信が開始されるまでに要した時間は 10 ミリ秒以下であった．

ハンドオフに関しては実通信端末を 3 台用意し，最初 2 端末間で通信していた一方の端末を，通信を継続したまま別の端末に切り替えることを確認した．



(a) cellular プログラムのユーザインタフェース



(b) ncell プログラムの実行状況

図 6.5.1 セッション層アーキテクチャの実装

6.6 関連研究

関連研究として，アプリケーションデータを送受信する通信チャネルの構築にあたり，別の通信チャネルを用いて構築する通信チャネルに関する情報をやりとりする技術についてまとめる．

まず，インターネットアプリケーションにおいてアプリケーションデータ用の通信チャネルとは別に通信チャネルを設けるプロトコルとして FTP [2] と SIP [3] について述べる．

FTP はアプリケーション内部でコントロールコネクションとデータコネクションを保持し，コントロールコネクションを通じて動的にデータコネクションを構築する機構を有している．FTP におけるコントロールコネクションの特徴は，コントロールコネクションは，クライアントプログラムからサーバプログラムに対して構築しなければならない点である．データコネクションの構築では，PORT コマンド，PASV コマンドを用いることで，コントロールコネクションのクライアント側，サーバ側のいずれからでもコネクションを構築することができるようになっている．しかし，これらのコマンドで通知する情報は，データコネクションのために開いたポート番号情報のみで，接続元の情報は送受信されない．また，コントロールコネクションとデータコネクションは，常に同じ通信デバイス上で動作しなければならない．したがって, データコネクションを終端する通信デバイスを変更する場合には，その通信デバイスとの間に再度コントロールコネクションを構築し，データコネクションを再構築しなければならない．

SIP もアプリケーション内部で制御用の通信チャネルとデータ送受信用の通信チャネルを保持するプロトコルである．SIP では，まず制御用通信チャネルをダイアログを用いて識別し，このダイアログに対応してデータ送受信用の通信チャネルが用意される．SIP では，FTP と異なり，制御用通信チャネルの構築の向きに制約はない．SIP において，データ送受信用通信チャネルの構築のためにやりとりされる制御メッセージ SDP [4] は，接続を待ち受ける IP アドレスとポート番号を通知するだけであり，接続元の IP アドレスとポート番号はつうちされない．これは，SIP がマルチキャストアドレスの広告プロトコルとして開発されたという経緯に依る．データ送受信用の通信チャネルを別の通信デバイスに切り替える際には，移動先の通信デバイスと SIP メッセージをやりとりする必要がある．

次に，データ通信とは別の通信チャネルを用いるモビリティサポートについて述べる．

ネットワーク層で実現するモビリティサポート [5, 6] は，アプリケーションプログラムに依存しないモビリティサポートであるため，必然的にデータ通信とは異なる通信チャネルでモビリティを実現している．しかし，ネットワーク層におけるモビリティサポートは，アプリケーションに対する透過性を保つため，データ通信を直接制御しているとは言えない．

トランスポート層で実現するモビリティサポートは，TCP をモビリティサポートの主たるターゲットにし，TCP を拡張することでモビリティを実現している [7]．すなわち，ひとつの TCP コネクションを用いて制御用情報とアプリケーションデータが流れる．さらに，TCP 自体がサーバプログラム，クライアントプログラムを明確に区別しているため，クラ

イアントプログラムからサーバプログラムに接続を行わなければ制御はできない．

アプリケーション層で実現するモビリティサポートとして，プロキシを用いたモビリティサポートがある．プロキシを用いるとエンドツーエンドの制御ができないため，エンドツーエンドで通信チャネルを制御することはできない．

6.7 むすび

本章では，ユーザが同時に複数のコンピュータを同時並行して利用する環境を想定し，3で述べたセッション層を拡張したセッション層アーキテクチャについて述べた．セッション層アーキテクチャでは，通信制御サーバを新たに導入し，セッション層制御部を実通信端末から分離した．また，通信制御端末を導入して，通信制御サーバのユーザインタフェースにするとともに，通信制御サーバと実通信端末間の制御通信を中継する機能を搭載した．

さらに，セッション層アーキテクチャを実装し，その動作を確認した．

参考文献

[1] Access Grid, "http://www.accessgrid.org/."

[2] J. Postel and J. Reynolds, "File transfer protocol (FTP)," RFC959, Internet Engineering Task Force, October 1985.

[3] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," RFC 2543, Internet Engineering Task Force, March 1999.

[4] V. Jacobson and M. Handley, "SDP: Session description protocol", RFC 2327, Internet Engineering Task Force, April 1998.

[5] C. Perkins, "Mobile IP," IEEE Communications Magazine, pp. 84-99, May 1997.

[6] M.Kunishi, M.Ishiyama, K.Uehara, H.Esaki, and F.Teraoka, "LIN6: A new approach to mobility support in IPv6", in Proceedings of International Symposium on Wireless Personal Multimedia Communication (WPMC) 2000, November 2000.

[7] A. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility. in Proceedings of 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp.155-166, August 2000.

7 セッション層情報を用いた通信資源管理

7.1 はじめに

本章では，セッション層が管理するフロー情報の通信資源管理への利用について述べる．

セッション層アーキテクチャではセッション層制御部がユーザの通信チャネルを制御するために必要なフロー情報を保持している．フロー情報とは，IP アドレス，ポート番号，トランスポート層プロトコルである．フロー情報はインターネットにおいて通信を行うための必要最小限の情報であり，また各通信のアプリケーションプロトコルを把握していなくてもフロー情報を用いるだけで簡単に通信を識別することができる．

一方で，インターネットにおける通信資源管理は重要性を増している．インターネットでは，インターネットに接続さえすれば，誰でもパケットをネットワークに送信することができるため，インターネット上の通信デバイスに攻撃を加えることが簡単である．さらに，インターネットはエンドツーエンドで動作するため，ネットワークが攻撃パケットとサービスパケットを識別することが困難であることが，問題をさらに顕著にしている．

そこで，インターネットを構成する各ドメインがセッション層が管理するフロー情報をユーザから受け取り，その情報に基づいてファイウォールを動的に制御することで，粒度の細かい通信資源管理の実現が可能となる．

以下では，まず 7.2 で通信資源管理におけるフロー情報の重要性を説明する．次に，7.3 で通信資源管理機構に求められる役割について述べる．そして 7.4 で実際に通信資源管理機構の設計を行い，それをふまえて 7.5 では，通信資源管理機構の一実現例として LAN における通信資源管理を実現するセキュリティゲートウェイの設計と実装について述べる．

7.2 通信資源管理とフロー情報

近年モバイル機器などの普及により，ユーザが複数の通信端末を使い分けることが多くなっている．また Wireless LAN の利用などユーザのネットワークへのアクセス形態も多様化している．このような状況において，共有デバイスやインターネットへの接続性をユーザに対して提供する管理者にかかる負担は増大している．

現在, ネットワークを介したコンピュータウイルスの蔓延や攻撃, 不正アクセスなど, ネットワーク利用における課題は数多く存在する．通信資源の管理者はこれらから通信資源を保護すると同時に，外部ネットワークに対する加害者となることがないようにユーザの通信資源利用を適切に管理しなければならない．

しかし現在一般に普及している通信資源管理機能を備えたシステムは，管理者やユーザの要求に対して柔軟に対応できているとはいえない．たとえばネットワークのセキュリティを守るはずのファイアウォールでは固定的なルールに基づくフィルタリングが行われているため, よく利用される特定のポートに対する攻撃を防ぐことは不可能であるほか, セキュリティを重視したフィルタリングルールではマルチメディアアプリケーションをはじめとする一部のサービス利用が制限されてしまう．一般に，ネットワークの管理ポリシーはネットワーク毎に異なるため，多様な管理ポリシーに柔軟に対応し，管理対象を自在に管理することのできる通信資源管理機構が求められている．

このような問題は近い将来ユビキタスコンピューティング環境が実現された場合，今以上にネットワークに接続される通信機器が増え，その形態も多様なものになるに従ってより顕著となり，管理者の負担が増加するであろう．

多様なネットワークポリシーに柔軟に対応し、管理対象を自由に管理することのできる通信資源管理機構の実現にあたりフロー情報は非常に重要である．フロー情報とは，アドレスとポート番号のペアおよびトランスポート層プロトコルの計五つのパラメータである．そして，これはインターネットにおいてアプリケーションプロトコルの詳細が分からなくても通信を簡単に区別することがアプリケーションに依存しない通信識別情報である

したがって，ネットワークを通過することが許されるフロー情報を通信資源の管理者が把握することにより，通信資源の管理を適切かつ柔軟に達成することが可能になる．たとえば外部からの不正に対しては，管理者が把握するフロー情報に合致しない通信フローをすべて不正なものとして排除することで対処できる. また内部からの不正やユーザ管理については, フロー情報とユーザ情報を関連づけることによって対応できる．

7.3 通信資源管理の課題

本節では通信資源の管理に関する問題点を整理し，通信資源管理に求められる機能について述べる．通信資源とは通信を行う際にユーザが直接的ないし間接的に使用する物理的なデバイスとソフトウェア資源，および各リンクにおいて占有する帯域を意味する．そして，通信資源管理の目的はこれら通信資源を管理，保護し，ユーザの快適で安全な通信サービス利用環境を維持することである．快適で安全な利用環境を構築するにあたっては解決すべき問題点が存在するが，以下ではそれら問題点を三つに大別して説明する．

・管理ネットワーク外部からの不正

通信資源管理における一つめの問題点は，外部ホスト，つまり管理者の管理管轄外のネットワークに接続しているホストからの攻撃である．インターネットでは誰もが自由にパケットを送り出すことができるため，悪意ある者が特定の対象に対して攻撃や総当たり攻撃などを行うこと自体を効果的に阻止することはきわめて困難である．

そのため，管理者はファイアウォールによるフィルタリングなどで，不正なパケットの内部ネットワークへの侵入を阻止することになる．しかし，ファイアウォールは外部ホストから送られたパケットが内部ホストのユーザにとって意図するものであるかを的確に判断することができないため，アドレスやポート番号を判断材料とする固定的なルールに基づくフィルタリングを行っている．一般にこのような固定的なルールによるフィルタリングでは，特定のポートが不特定多数のホストに対して常時開放されてしまうため，そのポートに対する

アタックなどからネットワーク内部のホストを守ることは困難である．また，セキュリティ維持のために用途の定まらないポートを閉じることもあるが，この場合動的なポート番号を使用するマルチメディアアプリケーションの利用が制限されることになる．このように，現在のファイアウォールは管理者にとっても，またユーザにとっても柔軟なフィルタリングが行えているとはいえない．

・管理ネットワーク内部からの不正

二つめの問題点は管理ネットワーク内部からの不正つまり，管理者が提供する通信資源の不正利用である．特に有線 LAN においては元々空間的な利用の制約があることから認証の重要性が軽視されており，ハブやスイッチ等の空きポートを悪用したネットワークの無断利用を防ぐ手だてはないのが現状である．

管理者が特定のユーザに対してのみ通信資源の利用を許可し，不正利用を排除したいと考えた場合，アドレスによるフィルタリングや RADIUS のような認証サーバを用いた IEEE 802.1x [1] によるユーザ認証などの方法がある．しかし，アドレスによるフィルタリングは端末識別情報による利用制限であるため，共有端末ではユーザを識別することができず、ユーザの個人端末を繋ぐ場合は端末識別の管理コストが増大する．また認証サーバの導入にはコストがかかるほか，ユーザはデバイスを利用する際に認証情報を入力する必要があり，一人

のユーザが同時に多端末を利用する状況では管理コストとユーザの負荷が大きい．

・正規ユーザの管理

通信資源管理における三つめの問題点は，正規ユーザによる通信資源利用状況管理の必要性である．これには大きく二つの意味があり，一つが他ネットワークへの加害行為の排除，もう一つがユーザ間の占有帯域の調整である．

まず，管理者は自らの管理する通信資源からウイルスの蔓延や外部ホストに対する攻撃を初めとする他ネットワークへの加害行為を速やかに検知し，それらを即座に止める責任がある．しかし現在の一般的な利用状況管理では，問題のあるパケットやフローの情報からユーザを即座に識別することは困難である．

また，VoIP をはじめとするマルチメディアアプリケーションやコンテンツ配信サービスではサービス利用において一定の帯域を維持することが求められ，サービス品質保証のための帯域制御の重要性が増している．このようなサービスを実現する上でも，フローとユーザの対応づけは必須である．

7.4 通信資源管理機構

7.4.1 アプローチ

多様なネットワークポリシーに柔軟に対応し、管理対象を自由に管理することのできる通信資源管理機構の実現に当たり、筆者らはフロー情報に注目した．フロー情報はアドレスとポート番号のペアおよびトランスポート層プロトコルの計五つのパラメータからなり，これはインターネットにおけるアプリケーションに依存しない通信識別情報である．また，管理者がユーザの行う通信のフロー情報を把握することで管理対象のネットワークを出入りするすべての通信フローの制御が可能になる．

つまり，通信資源の管理機構がすべてのフロー情報を一括して管理することにより，7.3で述べた通信資源管理上の問題点を解決するための機能を統一的に実現することができる．まず，外部からの不正に対しては，機構がもつフロー情報に合致しない通信フローをすべて不正なものとして排除することで対処できる．また内部からの不正やユーザ管理については，フロー情報とユーザ情報を関連づけることによって，利用者の不明な通信を拒否できるほかネットワーク利用における責任の所在をはっきりさせることでネットワーク内で発生する偶発的もしくは意図的なトラブルの解決も容易になる．

このように，フロー情報を用いることでユーザやフローを基本単位とする柔軟な通信資源管理を実現でき，多様な管理ポリシーに対応することができる．

ここで，これまで筆者がユーザ主体のネットワーキング実現に向け検討を進めてきたセッション層アーキテクチャがフロー情報を保持していることから，本稿では通信資源管理機能の実現にあたりセッション層アーキテクチャとの連携による解決を図る．具体的な通信資源管理機構の設計に先立ち，次節ではセッション層アーキテクチャにおいてフロー情報が果たしてきた役割について述べる．

7.4.2 セッション層アーキテクチャにおけるフロー情報の役割

セッション層アーキテクチャでは，ユーザは通信サービス利用において最初に通信相手とネゴシエーションを行う．この通信相手とは，たとえば利用するサービスがビデオチャットである場合には文字通り会話をする相手であり．ストリーミングによる音楽配信サービスを利用する場合には音楽配信サービスを提供するオンラインストアである．通信開始時のネゴシエーションは実通信端末間で直接通信をすることなく通信制御端末と通信制御サーバを介して行われる．このネゴシエーションによって，実通信において使用するアドレスとポート番号のペアおよびトランスポート層プロトコル，すなわちフロー情報が決定される．

実通信端末は実通信端末間の通信を開始する前にネゴシエーションによってフロー情報を得ることができ，フロー情報に合致する接続のみを受け付けることが可能になる．また，ユーザがアプリケーションサービスを利用している間，通信制御サーバがフロー情報を保持しているためユーザは通信制御端末を使って自分の行っている通信を統一されたインターフェー

スの元で管理できる．

このようにセッション層アーキテクチャでは通信制御サーバがユーザの行う通信のフロー情報を一括して管理している．次節では通信資源管理機構においてこのフロー情報をいかに利用するかについて述べる．

7.4.3 システム概要

通信資源管理機構は通信制御サーバの管理するフロー情報を必要とする．しかし，通信制御サーバは通信資源管理機構が管理すべき実通信端末のネットワークとは別のネットワークに設置されていることが一般的である．そのため，通信資源管理機構は通信制御サーバからフロー情報を入手する必要がある．また，一般にネットワークの管理ポリシーはネットワークごとに異なり，管理者が通信資源管理機構に対して求める機能も様々である．これらをふまえ，本機構では通信資源の管理者がそれぞれの管理ポリシーに基づき，必要な機能をもつ通信資源管理サーバを管理ネットワーク内に設置する．

通信資源管理サーバは通信制御サーバとの通信モジュールと 7.3 で述べた問題点を解決するための機能モジュールを持つ．通信モジュールは通信制御サーバとの間の制御メッセージに含まれる認証情報に基づいたユーザ認証を行い，認証を経た後に機能モジュールがフロー情報に基づき通信資源管理を行う．

その動作はきわめて動的であり，ユーザが今まさに行おうとしている通信に対するオンデマンドな要求に対して柔軟な対応ができる．また，ユーザ認証によってユーザを識別するため，ユーザ単位での粒度の細かい管理が可能となる．

7.4.4 制御要求の実現形態

以下ではまず，通信資源管理に必要なフロー情報がセッション層アーキテクチャにおいてどこで保持されているか，という観点から，セッション層アーキテクチャから通信資源管理機構への制御要求に関する実現形態を探る．

セッション層アーキテクチャでは通信制御サーバによってフロー情報が管理され，そのフロー情報が実通信端末に伝えられることによってアプリケーションサービスにおける通信が開始される．そのため，セッション層アーキテクチャでは通信制御サーバと実通信端末がフロー情報を保持しており，潜在的にはどちらからでも通信資源管理機構に対する制御要求を行うことが可能である．以下では，制御要求を通信制御サーバから行う場合と実通信端末から行う場合について，制御要求に必須な認証情報の安全性と遅延時間の観点から比較，検討を行う．

・認証情報の安全性

制御要求におけるセキュリティを考えた場合，制御要求に含まれる認証情報の流出をいかにして防ぐかが問題となる．認証情報の流出は，管理者にとって不正なユーザの通信資源利用を許してしまうだけでなく，ユーザに対して金銭的被害を与える可能性がある．

まず，実通信端末から通信制御要求を行う場合については，認証情報を実通信端末に入力すること自体に危険性がある．これは，ユーザ自身の所有物ではなく共有端末を実通信端末として一時的に利用する場合，ユーザが悪意ある者の設置した端末を使用してしまい認証情報を盗まれる可能性があるためである．認証情報をネットワークに流す際に暗号化を施したとしても，認証情報の入力段階でスパイウェアやキーロガー等によって情報が盗まれることに対しては何の効力もない．これに対して，通信制御サーバから制御要求を行う場合にはあらかじめ認証情報を通信制御サーバに蓄えておくか，あるいは通信制御端末上で入力する方法が考えられる．通信制御端末は携帯電話のようなユーザ専用の端末であり，いずれの場合においても信頼できない端末に対して直接認証情報を入力することがないため安全である．

次に，認証情報を暗号化するための鍵交換について考える．ネットワーク内に設置された実通信端末からの制御要求では，実通信端末と通信資源管理サーバの間で事前に鍵交換を行っておくことが可能である．一方，ユーザが持ち込んだ実通信端末からの制御要求と通信制御サーバからの制御要求では，実通信端末の行うサービス広告に通信資源管理サーバの公開鍵を含めることで，こちらも通信資源管理サーバに対して安全に制御要求を行うことができる．

また，通信資源利用に対する課金モデルの構築を考えると，携帯キャリアやプロバイダによって管理される通信制御サーバから制御要求を行う場合において，ユーザの通信資源利用に対する代理課金や通信制御サーバ間の連携によるローミングサービスの展開が可能である．

・遅延

次に，制御要求にかかる遅延時間の検討を行う．セッション層アーキテクチャでは，フロー情報が通信制御サーバによって決定された後に実通信端末に伝えられるため，通信制御サーバが直接ネットワーク資源管理制御機構に対して制御要求を行った場合の方が遅延は少ない．加えて，通信制御サーバから制御要求を行った場合には，通信制御サーバ間のネゴシエーションによって制御要求の完了を確認した上で，実通信端末に対して通信開始要求を行うことが可能である．それに対し，実通信端末で制御要求を行う場合には，通信相手側の制御要求の完了を確認できないまま通信を開始するため，通信相手がファイアウォールのフィルタリング設定の変更等が完了していない可能性があり，通信路が確立される前に送られたデータが相手に届かないという問題がある．セッション層アーキテクチャのプロトコルを拡張することにより，実通信端末から制御要求を行った場合でも通信相手側の制御要求の完了を事前に確認することは可能であるが，通信を開始する前に必要なネゴシエーションの手順が増えるため遅延時間が増大する．

7.4.5 管理機能の実現手法

以下では通信制御サーバから制御要求を行うモデルについて，本機構における制御要求を用いた通信資源管理機能の具体的な実現手法について述べる．

・フロー情報を用いた動的なフィルタリング

管理ネットワーク外のホストによる攻撃からの保護はフロー情報を用いた動的なフィルタリングによって実現される．まず初期状態として，管理ネットワークのゲートウェイを通過する通信を内部から外部，外部から内部ともにすべて禁止する．そして，通信制御サーバからの制御要求を受理した通信資源管理サーバが，フロー情報に基づいたフィルタリングルールの動的変更を行うことでユーザが行う特定の通信を可能にする．フロー情報では，内部ホストの IP アドレスとポート番号だけでなく，通信相手の IP アドレスとポート番号，およびトランスポート層のプロトコルが指定され，フィルタリングルール変更の際にはこれらすべての情報を使用する．またユーザの通信が終了した場合，再度通信制御サーバが制御要求を行い，通信開始時に設定したフィルタリングルールを削除する．これによって，外部ネットワークとの通信時に発生するセキュリティ上の危険性は最小限にとどまり，内部ホストを DoS アタックなどの攻撃から守ることも可能になる．また，ポート番号のみに基づいたフィルタリングではないため，セキュリティのためにアプリケーションサービス利用が制限されることもなく，固定的なフィルタリングルールの設定が不要であることから管理者の負担も軽い．

なお，本機構ではフィルタリングをフロー情報に基づいて行うため，IP アドレスとポート番号を偽装されたパケットはファイアウォールを通過してしまう．しかし，セッション層アーキテクチャではサーバアプリケーションを含め動的なポート番号を使用するため，利用されているポート番号を推定しパケットを偽装することはきわめて難しい．

・制御要求におけるユーザ認証および帯域制御

制御要求時に行われるユーザ認証において正当な認証情報を持たない制御要求は排除されるため，ネットワーク内部からの不正なネットワークアクセスも同時に拒否される．これにより，管理者の意図しないユーザによるネットワーク利用を防ぐことが可能である．同時に，ユーザの利用状況をシステムや管理者が把握することも容易になり，ユーザ間の占有帯域調整をフローレベルで実現できる．また，ユーザ認証と連携した課金モデルの導入により，特定のユーザの通信を優先させることや，ユーザの要求に基づいて特定のアプリケーションにおける通信を優先させるなどといった商用サービスにも展開できる．

7.5 セキュリティゲートウェイ

本節では通信資源管理機構の一例として，LAN における通信資源管理を実現するセキュリティゲートウェイの設計と実装，およびその基本的な性能評価について述べる．

7.5.1 設計

通信資源管理機構は，セッション層アーキテクチャが動作する場合に汎用的な通信資源管理機能を提供することが可能である．ここでは，7.4 で述べた通信資源管理機構を LAN に適用した通信資源管理サーバ（セキュリティゲートウェイ）について述べる．

セキュリティゲートウェイの提供する通信資源管理機能は二つある．一つめは，フィルタリング精度の高い柔軟なファイアウォールによって，外部ホストからの攻撃に対しネットワークのセキュリティを高めることであり，二つめは，ネットワーク内部のホストからのネットワークアクセスを制御することである．セキュリティゲートウェイは実通信端末から構成される内部ネットワークとインターネットに接続された外部ネットワークの境界に設置され, 制御要求を受け付けることでその役割を果たす. セキュリティゲートウェイはインターネットカフェや無線 LAN ホットスポットなどの商用サービスにおいても有効であり, セキュリティゲートウェイにおける認証・課金処理によって統一的なサービス管理を行うことができる．またユーザにとっても，セッション層アーキテクチャを利用することで，認証情報を共有端末に直接入力する必要が無くなりセキュリティが向上する．

7.5.2 実装

セッション層アーキテクチャおよびセキュリティゲートウェイからなるシステム全体の構成を図 7.5.1 に示す．実通信端末および通信制御端末に関しては Windows XP 上で開発，動作確認を行った．また，通信制御サーバおよびセキュリティゲートウェイについては，FreeBSD 5.3 上で開発を行った．セキュリティゲートウェイではフィルタリングルールの変更に際して FreeBSD における IPFW を利用しており, 動作確認も同じく FreeBSD 上で行った．

7.5.3 開発ソフトウェア

ここではセキュリティゲートウェイ上で動作する SG_FIREWALL について述べる．SG_FIREWALL はサーバプログラムとして動作し，通信制御サーバからの制御要求を受け付ける．制御要求を受け取るとまず制御要求に含まれる認証情報を SG_FIREWALL があらかじめ保持しているユーザの認証情報と照合しユーザ認証を行う．ユーザ認証を経た後その制御要求を受理するために必要な IPFW の適用ルールをフロー情報を参照することによって作成する IPFW においてフィルタリングルールを追加．削除する際の書式はそれぞれ以下のようになっている．

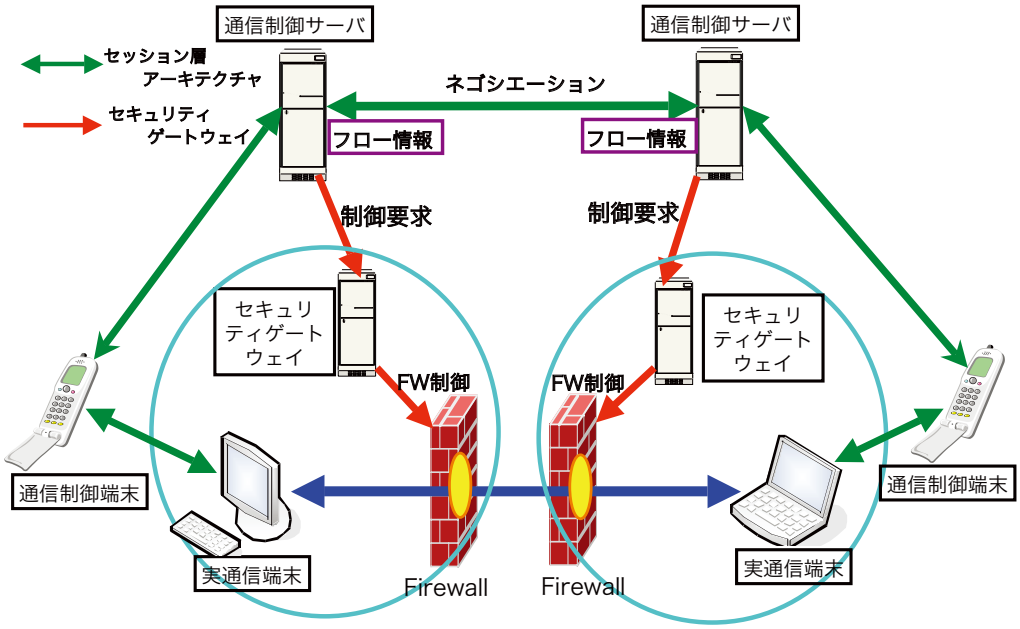


図 7.5.1 セキュリティゲートウェイの構成

ipfw add ルール番号トランスポート層プロトコル
　from 送信元　アドレス送信元ポート番号
　to 送信先　アドレス送信先ポート番号
ipfw delete ルール番号

IPFW におけるルール番号は本来，フィルタリングルールの優先順位を決定するものであるが，セキュリティゲートウェイでは，一度設定したルールを後で削除する際に使用する．そのため SG_FIREWALL は内部にテーブルを持ち，現在適用しているフィルタリングルールをルール番号とともに管理する．適用するルールを作成するとプログラム内部から IPFW を呼び出し，IPFW に対して新しいルールの追加を行う．

7.5.4 性能評価

ここでは，今回作成したセキュリティゲートウェイの簡単な性能評価として，制御要求に要する遅延時間の測定を行った．制御要求にかかる遅延時間は三つの要因からなる．一つは通信制御サーバとセキュリティゲートウェイ間の RTT であり両者のネットワーク的な距離に依存する．二つめは，制御要求を受け取ってから IPFW に適用するルールの作成に要する時間である．最後に三つめの要素は，IPFW の実行に要する時間である．

ここでは，ルール作成と IPFW の実行にかかる時間のスケーラビリティに注目し測定を行った．測定結果を図 7.5.2 に示す．図からルール作成に要する時間は保持ルール数が増加するに従ってほぼ線形に増大していることがわかる．これは，新しいルールに対するルール番号を決定する際に保持ルールに対して線形探索を行っていることによる．測定結果から通常の利用環境では制御要求に要する遅延時間は十分に実用可能な値であるといえるが，保持ルール数が数千以上に達するような環境では，ルール番号決定処理により効率的なアルゴリズムを利用することで遅延時間の改善が可能である．

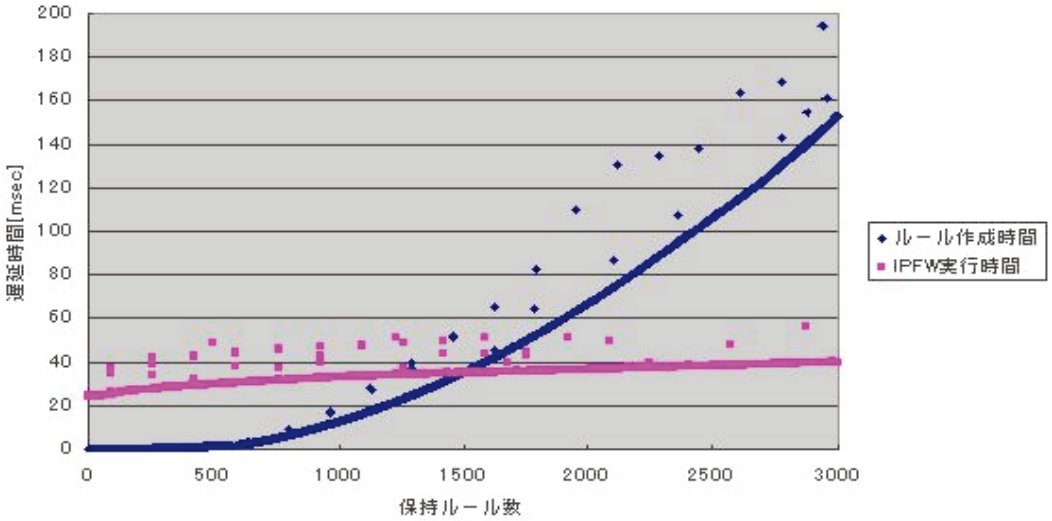


図 7.5.2 セキュリティゲートウェイの性能評価

7.6 むすび

本章では通信資源の管理を統一的におこなう通信資源管理機構に求められる役割を明確にし，セッション層アーキテクチャを用いた通信資源管理機構の実現手法について述べた．本機構はセッション層アーキテクチャの管理するフロー情報を利用し，柔軟かつ安全な通信資源管理を行うことを目指したものである．また本章では通信資源管理機構の一例としてインターネットを構成する各ドメインのファイアウォールにおいて通信資源管理を行うセキュリティゲートウェイの設計と実装についても説明を行った．セキュリティゲートウェイは，従来のファイアウォールよりも粒度の高いフィルタリングを行い，同時にユーザ認証によってネットワークアクセス制御を実現している．

参考文献

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," RFC 2543, Internet Engineering Task Force, March 1999.

[2] Institute of Electronic and Electrical Engineers (IEEE), "Port based network access control," Standard 802.1x, 2001.

8 セッション層を用いたサービス

8.1 はじめに

本章では, セッション層アーキテクチャの利用例を紹介する．本章では, セッション層アーキテクチャを用いた遠隔会議システムの設計と実装について述べる．

以下, 8.2 では遠隔会議システムの現状について述べ, 8.3 でセッション層アーキテクチャを用いた遠隔会議システムについて述べる．8.4 で遠隔会議システムの実装について述べる．

8.2 遠隔会議システムの現状

現在の遠隔会議システムは主に導入時の要求に応じてシステムを設計し，比較的長期間にわたり同じシステムを利用し続けている．しかし，近年のコンピューティング技術の発展により容易に扱えるようになったマルチメディアを既存のアナログベースの遠隔会議システムに組み込むことは非常に困難である．例えば，プレゼンテーションスライドを遠隔会議先に提示するといった非常に簡単なことでも，遠隔会議システムに組み込むには新たなシステムの構築を要する場合が多い．これは，多くの遠隔会議システムにおいて拡張性を確保して設計することが困難であることを示している．

このような背景のもと，近年注目を集めているのが SIP [1] を基盤とした遠隔会議システムである．SIP はインターネットマルチキャストを使った遠隔会議のメディア情報の広告プロトコルを起源としたアプリケーションメッセージングフレームワークである．SIP を基盤とした遠隔会議システムでは SIP のメッセージに対応したアプリケーション同士であれば，比較的簡単にシステムを拡張することが可能となる．しかしながら，SIP はそれぞれのアプリケーションでメッセージ処理を行うプロトコルであるため，SIP アプリケーションの開発者はメッセージ処理部や通信相手への接続部，マルチメディアデータ処理部をすべて開発しなくてはならない．さらに，システムの拡張性を考慮するとベンダ間の互換性の確保が重要であり，アプリケーションプロトコルである SIP においては重要な問題となる [2].

一方，セッション層アーキテクチャを用いた遠隔会議システムでは，共通のミドルウエアによって通信制御を実現するため SIP ほど互換性の問題は発生しにくい．さらに，アプリケーションとネットワークは通信中を除いて常に分離された状態にあるためセキュリティの問題も回避できる．すなわち，アプリケーション開発者はインターネットからの攻撃や通信制御をあまり気にすることなく，本来のマルチメディアデータ処理を重点的に行うことが可能となる．以下では，セッション層アーキテクチャを用いた遠隔会議システムについて述べる．

8.3 セッション層アーキテクチャを用いた遠隔会議システム

8.3.1 遠隔会議システムにおける通信制御

遠隔会議において多様なマルチメディアリソースを利用可能な場合，ユーザはネットワーク越しに相手のリソースを自在に制御することで，会議の進行を滞らせることなく臨場感の高い会議を進めることが可能になる．遠隔リソースの自在な制御とは，例えば，通信相手を写すカメラを切り替える，またはカメラからプレゼンテーションスライド画面に切り替えるといった制御である．筆者らは，多様なマルチメディアリソースを扱う遠隔会議システムにおける特徴が遠隔リソースの自在な制御にあることに注目し，遠隔会議システムを設計した．以下では，カメラ，マイク等のマルチメディアリソースを会議リソースと呼ぶ．

上記の遠隔会議システムを実現するには，下記の三点が必要である．

(1) 通信相手に制御を許可する会議リソース（公開リソース）の決定

(2) 会議中に限った公開リソースの自由な制御

(3) 会議参加者に限定した公開リソースの制御

これらの要件を利用例で示すと以下のようになる（図 8.3.1）．
ユーザ A は複数のカメラの中から，カメラ 1 とカメラ 2 を会議リソースとして選択した．他のカメラは公開されない．公開後，ユーザ A と遠隔会議を行うユーザ B は，ユーザ A の操作を必要とせずに公開されているカメラ 1 の画像をユーザ B の手元のディスプレイに表示することが可能となる．公開されているカメラ 1 とカメラ 2 にアクセスできるのはユーザ B だけである．

8.3.2 セッション層アーキテクチャを用いた遠隔会議システム

8.3.1 で述べた要件を満たすシステムとして，通信制御を通信相手側が遠隔から安全に行う機構を構築した．具体的には，通信制御機能を通信相手からでも制御可能にするプロキシソフトウエアを，セッション層アーキテクチャを利用した通信アプリケーションとして作成した．セッション層アーキテクチャを用いることで，会議中に限り通信制御機能を通信相手に委譲することが可能 (2) になるだけでなく，会議参加者に限定した制御も可能 (3) になる．

作成したソフトウエアは以下のように動作する．まず，ユーザ A が公開を許可するリソースを選択する．次にセッション層アーキテクチャを用いて発呼処理をおこない遠隔会議を開始（プロキシソフトウエア間の通信が確立）する．開始後，ユーザ A が公開を許可したリソースの一覧をユーザ B 側に提示し，ユーザ B がリソースを選択すると，プロキシソフトウエアがあたかもユーザ A がそのリソースを選択しユーザ B との通信を希望しているかのように動作する．その後，ユーザ B は，通信制御サーバを介してユーザ A からの通信開始要求が届くが，これはあらかじめユーザ B が希望したものであるため自動的に通信が開始されることになる．

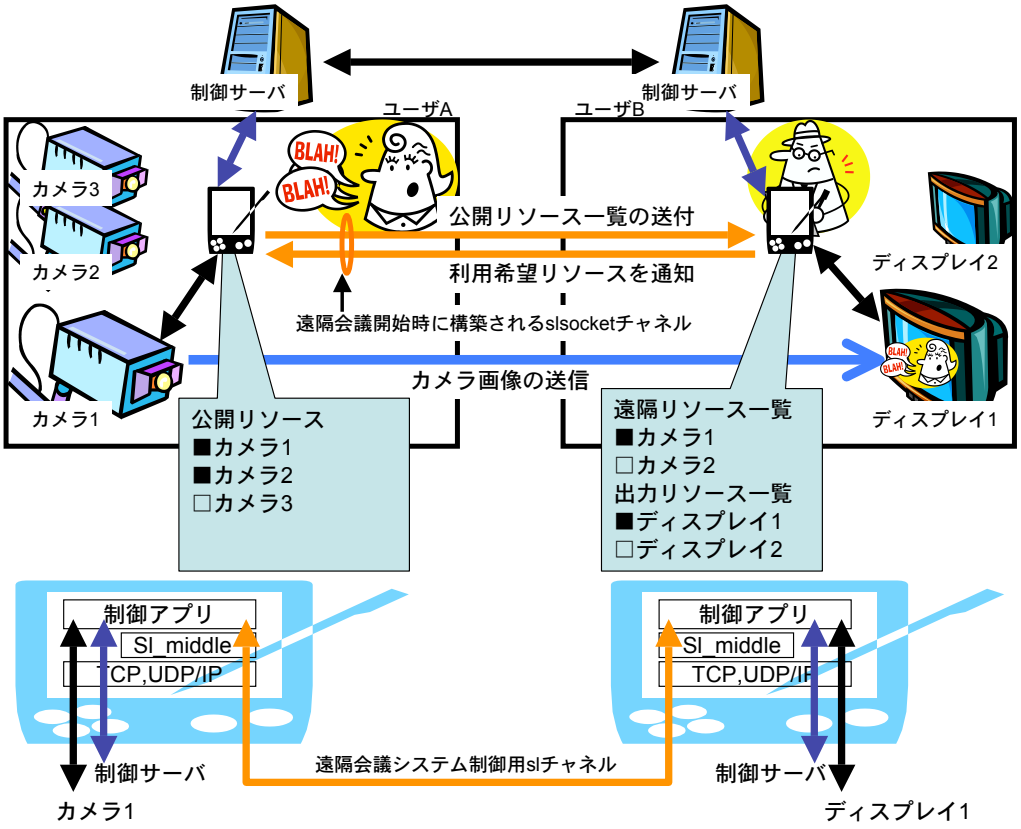


図 8.3.1 セッション層アーキテクチャを用いた遠隔会議システム

8.4 実装

筆者は，8.3 で述べた通信制御機能を通信相手側に委譲する機構の実装を行うとともに，DVTS[3] を基盤とした遠隔会議システムを構築した．

8.4.1 通信制御機能の委譲

通信相手に委譲する通信制御機能は，通信制御をより使いやすくするために豊かなユーザインタフェースを構築できることが望ましい．そこで HTTP を用いてユーザインタフェースを提供できる機構を実装した．

本機構は，HTML 等で記述された情報がセッション層 API を通して通信相手に提示され，通信相手の要求は HTTP に従ってセッション層 API を通して伝えられる．さらに，その要求に従って動的に HTML が作成され再送信される．なお，複数の TCP コネクションを利用する HTTP をセッション層 API を通して利用するために，ポートフォワーディングを用いている．

通信制御機能の委譲は，通信相手側の制御情報表示部（ブラウザ），sl_wwwclient，sl_wwwproxy，WWW プロキシ，制御情報生成部によって構成される．sl_wwwclient はユーザが利用するブラウザと同じ端末で動作し，ブラウザのアプリケーションデータを sl_wwwproxy にポートフォワーディングする．sl_wwwproxy はプロキシアプリケーションと同じ端末で動作し，sl_wwwclient からポートフォワーディングされてきたそれぞれの通信フローを分離して WWW プロキシを介して制御情報生成部に転送する．sl_wwwclient と sl_wwwproxy の構成を図 8.4.1 に示す．

以下では，実装の詳細を述べる．ブラウザを Windows XP 上で動作させ，WWW プロキシを NetBSD 2.1 上に実装した．本実装では WWW プロキシアプリケーションに squid を，ポートフォワーディングに SSH を用い，Internet Explorer のブラウザコンポーネントを用いた．また，通信制御情報制御部には，Apache を用いた．作成したソフトウエアは以下のように動作する．まず，sl_wwwclient と sl_wwwproxy の間の通信路をセッション層アーキテクチャに基づいて確立する．通信路が確立された後，sl_wwwclient は SSH によるポートフォワーディングを行う．なお，SSH は公開鍵認証が行えるようにあらかじめ，sl_wwwclient の公開鍵を sl_wwwproxy 側に登録している．これにより，ブラウザのプロキシの設定を SSH によってポートフォワーディングされたポートにすることで，HTTP をセッション層 API を通して行うことができるようになる．なお，ブラウザが利用するプロキシポート番号は動的に決定され，ブラウザ起動時に自動的に設定される．

8.4.2 DV を用いたマルチメディア送信

DV を遠隔会議システムの基盤とした理由は，高画質，高音質であるため遠隔会議をストレスなく実現することができるからである．DV の送受信システムには DVTS [3] を用いた．本来であれば，DVTS をセッション層アーキテクチャに適合するように改変するのが望まし

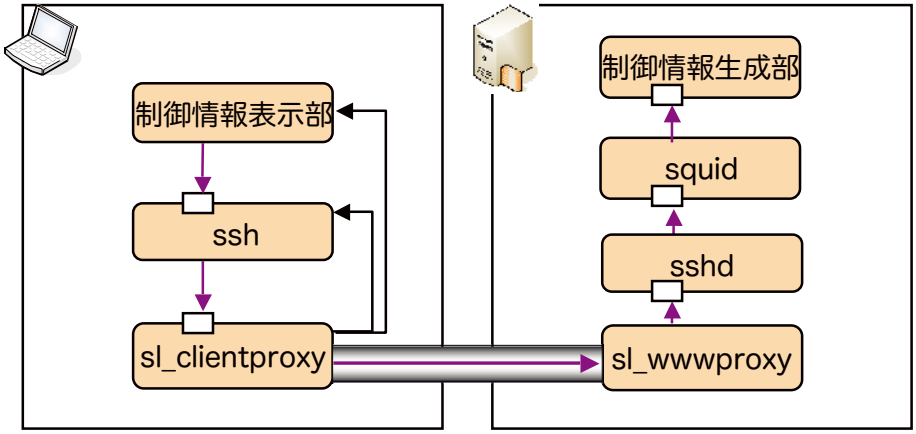


図 8.4.1 sl_wwwclient と sl_wwwproxy の構成

いが，DVTS の内部構造に触れることなく簡単に実現するため，また DVTS 自体の改変作業等に依存しないようにするため，DV 送信側，受信側にセッション層アーキテクチャを用いたプロキシソフトウエアを作成した．このプロキシソフトウエアは，DV 送信側において dvsend プログラムから UDP フレームを受け取りセッション層 API を用いたインタフェースにデータを書き出す sl_dvsendporxy と DV 受信側においてセッション層 API を用いたインタフェースからデータを受け取り UDP フレームにして DV 受信側の dvrecv プログラムにデータを書き出す sl_dvrecvproxy で構成されている．すなわち，セッション層アーキテクチャを用いた DV 送信は，内部的に dvsend と sl_dvsendproxy の 2 つのプログラムで構成されている．DV 受信側も同様である．

8.4.3 動作確認

上記の遠隔会議システムの開発後，柏キャンパスと本郷キャンパスにそれぞれ DV 機材を設置し，セッション層アーキテクチャを用いた遠隔会議システムの動作確認を行った．

8.5 むすび

本章では，セッション層アーキテクチャを用いた DVTS を基盤とした遠隔会議システムの設計と実装について述べた．

参考文献

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," RFC 2543, Internet Engineering Task Force, March 1999.

[2] VoIP/SIP 相互接続検証タスクフォース , http://www.jpnic.jp/ja/voip-sip-tf/

[3] DVTS Digital Video Transfer System, http://www.sfc.wide.ad.jp/DVTS/

9 結論

9.1 本論文の成果

本論文では，インターネットの相互接続性を最大限生かしユーザが主導的に通信を制御しインターネット上で自由自在に情報を操れるようなインターネットの共通サービスプラットフォームをアーキテクチャ的観点から考察し，これからのインターネットに必要なセッション層の構成について論じている．

具体的には，まずユーザ主導型通信制御の意義について示し，通信制御を IP アドレス，ポート番号，トランスポート層プロトコルを用いて行うセッション層の必要性を述べている．そして，セッション層を用いたサービスモビリティを実現するための認証技術を示している．また，IEEE 802.11 省電力モバイル端末を例にセッション層が管理する通信情報の範囲を明らかにしている．さらに，セッション層を遠隔から制御するセッション層アーキテクチャを提示し，セッション層が管理する通信制御情報をネットワーク管理に応用する通信資源管理機構を示している．最後に，セッション層を用いたアプリケーションサービスシステムを示している．

以下に，本論文で得られた主たる成果を述べる．

・ユーザ主導型通信制御の重要性を明らかにし，インターネットとユーザ主導型通信制御の親和性を示した．ユーザ主導型通信制御は，ユーザがアプリケーションプログラムを制御するのではなくアプリケーションプログラム間の通信チャネルを制御する機構である．また，ユーザ主導型通信では通信チャネルの制御権が通信を行う両ユーザにあることを明確に示した機構である．

・ユーザ主導型通信制御をインターネットで実現するためにセッション層の設計を行った．セッション層の設計指針はインターネットとの親和性，ユーザ主導性，アプリケーションプログラムの多様性の確保にある．とくに，ユーザ主導性を確保するために，ユーザ間の認証処理とアプリケーションプログラムの制御権を明確に分離している．

・セッション層の実装を行うとともに実装を用いて評価を行い有効性を示した．セッション層の実装では，アプリケーションプログラムが利用する API とセッション層機能を実現するミドルウエアを設計し，セッション層の実現方法を明らかにした．

・セッション層を用いてサービスモビリティを行う際の認証処理を示した．サービスモビリティでは通信デバイスが変更になることもあり，key-insulated 公開鍵暗号方式を用いることで通信の安全性を維持する手法を示した．

・セッション層が管理する通信情報は IP アドレス，ポート番号，トランスポート層プロトコルで十分であることを示した．

・セッション層を遠隔から制御するセッション層アーキテクチャを示した．セッション層アーキテクチャは，通信の管理をユーザごとにまとめた上で固定端末で制御するため，複数の通信デバイスを同時に利用するときの利便性と安全性を高めるアーキテクチャである．

・セッション層アーキテクチャを実装し，その有効性を遠隔会議システムとして示した．セッ

ション層アーキテクチャの実装では，遠隔制御にセルラーネットワークとセルラー端末を用いることで，ユーザの移動に制約されない通信管理が実現できることを示している．

以上に示すように，本論文はインターネットにおけるユーザ主導型通信制御の必要性，ユーザ主導型通信制御を実現するために必要となるセッション層の導入とその構成を示している，本論文では，これらの手法を提案すると同時に，実装実験を通してその有効性を実証している．

9.2 今後の展望

今後の展望として，インターネットにセッション層を導入した際の課題を示すとともに，インターネット以外のネットワークにおける通信制御の必要性について述べる．

インターネットにセッション層を導入した際に課題となるのは，セッションダイアログサービスとプレゼンテーション層機能の欠落である．

セッションダイアログサービスとは，セッション層が構築する通信チャネルを流れるデータの同期ポイントを示す機能である．セッション層を用いると，アプリケーションプログラムに対して通信チャネルの切り替えは完全に隠蔽される．したがって，通信チャネルの切り替えにおいて発生するデータロスをアプリケーションプログラムが把握することはできない．すなわち，セッション層を介して流されるデータはブロック化され，アプリケーションプログラムがデータの開始点を把握できるようにしなければならない．

プレゼンテーション層の機能は，通信チャネルの切り替えに伴って受信するデータの品質の変更等を行う機能である．例えば，セルラー端末で映像情報を受信していたユーザが通信チャネルを広帯域ネットワークに接続された大型ディスプレイに切り替えた場合は，大型ディスプレイに適したコンテンツが望ましい．すなわち，通信チャネルをユーザが切り替えた際のコンテンツアダプテーションを行うための機能が必要となる．

セッションダイアログサービスとプレゼンテーション層の機能は組み合わせて実現することが可能であり通信の効率の観点からも望ましいであろう．なお，これらの機能を階層として設け，アプリケーションプログラムとは独立に構成することについては階層が増加することによって通信のオーバヘッドが発生するため，筆者は否定的である．アプリケーションプログラムが選択的に利用できるライブラリの形での機能追加が期待される．

インターネットにセッション層を導入後により要求が高まるのは QoS ルーティングであろう．セッション層が管理する通信情報はフローであり，フロー毎に QoS パラメータを設けることで，ユーザはより豊かにネットワークを利用することが可能になる．現在，QoS の確保は主に回線交換的アプローチによって実現しており，セッション層のエンドツーエンドの思想と相容れない．この差異をどのように縮めるかが研究を進めていく上で重要になろう．

インターネット以外のネットワークにおいても通信制御が必要になろう．本論文で述べたセッション層は IP ネットワークを前提にしているが，Ethernet で構築されたローカルエリアネットワークにおいても通信制御は必要である．Ethernet は元来ブロードキャストネットワークであり，このネットワークには多数の機器が接続されている．これらの機器をユーザが必要に応じて選択的にグループ化し安全性を確保する通信制御機構が求められるであろう．

発表文献

論文誌

金子晋丈 , 中山雅哉 , 市川雄一 ,
" オンデマンド時刻取得に基づく高精度なネットワーク時刻同期プロトコルの実装と評価 ,"
電子情報通信学会論文誌 ,(投稿中).
金子晋丈 , グエンホアイソン , 橋本洋平 , ダムラクス タナプーム , 森川博之 , 青山友紀 ,
" 省電力 IEEE 802.11 モバイル端末の TCP スループットの改善 ,"
電子情報通信学会論文誌 ,(投稿中).

学会誌

中内清秀 , 金子晋丈 , 齊藤昭 , 森川博之 ,
" ディジタルビデオを用いたリアルタイムマルチメディア通信システム ,"
情報処理学会誌 , vol. 42, no. 12, pp. 793-798, Aug. 2001.

査読つき国際会議

N. Imai, K. Kaneko, H. Morikawa, and T. Aoyama,
"On-demand Data Prefetching System for Spotted Access Networks,"
In Proc. Seventh Asia Pacific Conference on Communications (APCC2001), pp.61-64, Tokyo, Japan, Sep. 2001.
K. Kaneko, H. Morikawa, and T. Aoyama,
"Session Layer Mobility Support for 3C Everywhere Environments,"
In Proc. of the 6th International Symposium on Wireless Personal Multimedia Communications (WPMC 2003), Vol.2 pp.347-351, Yokosuka, Japan, Oct. 2003.
N. Hoaison, A. Shindo, K. Kaneko, H. Morikawa, and T. Aoyama,
"Personal Mesh: A Desgin of Flexible Internet Access for Personal Area Network,"
In Proc. of the 6th International Symposium on Wireless Personal Multimedia Communications (WPMC 2003), Vol.2 pp.356-360, Yokosuka, Japan, Oct. 2003.

シンポジウム、ワークショップ

金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 ,
" アクセスリンク帯域を考慮した TCP フロー制御 ,"
マルチメディア、分散、協調とモバイルシンポジウム (DICOMO2001), Naruto, Japan, Jul. 2001.
情報処理学会 モバイルコンピューティングとワイヤレス通信研究会 (MBL) 優秀論文
金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 , 中山雅哉 ,
" 動的なインターネットサービスのためのセッションレイヤモビリティサポート ,"
第 4 回 YRP 移動体通信産学官交流シンポジウム , Yokosuka, Japan, Jul. 2002.

今井尚樹 , 金子晋丈 , 森川博之 , 青山友紀 ,
" ユビキタスアプリケーション実現に向けたサービスハンドオフ機構 ,"
第 4 回 YRP 移動体通信産学官交流シンポジウム , Yokosuka, Japan, Jul. 2002.

S. Lim, K. Kaneko, H. Morikawa, and T. Aoyama,
"Secure Session Migration Using Key-Insulated Public-Key Cryptosystems,"
In Proc. of the 1st Joint Workshop on Mobile Multimedia Communicatins (JWMMC2003), pp.57-61, Seoul, Korea, Jul. 2003.
金子晋丈
“ユーザ主導型通信を実現するセッション層アーキテクチャ ,”
大域ディペンダブル情報基盤シンポジウム 2005, Tokyo, Japan, Mar. 2005.
斉藤哲也 , 金子晋丈 , 森川博之 , 青山友紀 , 羽鳥光俊 ,
" セッション層アーキテクチャにおける実通信機構の設計および実装評価 ,"
マルチメディア、分散、協調とモバイルシンポジウム (DICOMO2005), Iwate, Japan, Jul. 2005.
J. Ok, S. Komorita, K. Kaneko, K. Gogo, R. Shibui, S. Fujimaki, K. Mitani, F. Teraoka, H. Morikawa, and T. Aoyama,
"Implementation of Fast Handover Scheme using L2 Trigger Mechanism in IEEE 802.11 Wireless LAN,"
第 7 回 YRP 移動体通信産学官交流シンポジウム , Yokosuka, Japan, Jul. 2005,

研究会、大会

金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 , 浅野欽也 , 徳田清仁 ,
" アクセスリンク帯域を考慮した TCP フロー制御 ,"
電子情報通信学会総合大会 , B-7-192, Kusatsu, Japan, Mar. 2001.
森川博之 , 鄭武龍 , 國頭吾郎 , 今井尚樹 , 金子晋丈 ,
" アプリケーションから見たワイヤレスインターネット ,"
電子情報通信学会ソサイエティ大会 , TB-1-5, Tokyo, Japan, Sep. 2001.
今井尚樹 , 金子晋丈 , 森川博之 , 青山友紀 , 浅野欽也 , 徳田清仁 ,
" スポット型ネットワークにおけるオンディマンド型データプリフェッチシステム ,"
電子情報通信学会総合大会 , B-5-147, Kusatsu, Japan, Mar. 2001.
今井尚樹 , 金子晋丈 , 森川博之 , 青山友紀 ,
" ユビキタス環境におけるサービスモビリティサポート ,"
電子情報通信学会総合大会 , B-7-13, Tokyo, Japan, Mar. 2002.
金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 , 中山雅哉 ,
" 多様化するインターネット環境のためのセッションレイヤモビリティサポート ,"
電子情報通信学会総合大会 , B-7-14, Tokyo, Japan, Mar. 2002.
河内佑介 , 金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 ,
" セッションレイヤモビリティサポートの実装とハンドオフ性能評価 ,"
電子情報通信学会総合大会 , B-7-15, Tokyo, Japan, Mar. 2002.
新藤晃浩 , 金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 ,
" モバイル CDN におけるセッションレイヤモビリティサポート ,"
電子情報通信学会総合大会 , B-7-63, Tokyo, Japan, Mar. 2002.
M. R. Jeong, K. Kaneko, Y. Kochi, N. Imai, G. Kunito, H. Morikawa, and T. Aoyama,
"Bringing flexibility into Ubiquitous Personal Networks,"

電子情報通信学会総合大会 , SB-12-5, Tokyo, Japan, Mar. 2002.
金子晋丈 , 森川博之 , 青山友紀 , 中山雅哉 ,
" 多様化するインターネット環境におけるエンドツーエンド型モビリティサポート ,"
電子情報通信学会技術研究報告 , MoMuC2002-8, Tokyo, Japan, May. 2002.
金子晋丈 , 河内佑介 , 森川博之 , 青山友紀 , 中山雅哉 ,
" セッションレイヤにおけるエンドツーエンド型モビリティサポートの実装と評価 ,"
電子情報通信学会技術研究報告 , MoMuC2002-9, Tokyo, Japan, May. 2002.
川原圭博 , 新藤晃浩 , 金子晋丈 , 森川博之 , 青山友紀 ,
" セッションレイヤモビリティサポートを用いた IP リモートコントロールカー ,"
電子情報通信学会ソサイエティ大会 , B-7-29, Miyazaki, Japan, Sep. 2002.
金子晋丈 , 今井尚樹 , 森川博之 , 青山友紀 , 中山雅哉 ,
" セッション層モビリティサポートによる端末間ハンドオフの実現 ,"
電子情報通信学会総合大会 , B-15-3, Sendai, Japan, Mar. 2003.
林素娟 , 金子晋丈 , 森川博之 , 青山友紀 ,
" セッション層モビリティサポートにおけるセキュアなハンドオフ機構 ,"
電子情報通信学会ソサイエティ大会 , B-15-14, Niigata, Japan, Sep. 2003.
ダムラクス タナプーム , 金子晋丈 , 森川博之 , 青山友紀 ,
"IEEE802.11b 基地局における TCP プロキシのスリープ間隔に関する性能評価 ,"
電子情報通信学会ソサイエティ大会 , B-15-3, Niigata, Japan, Sep. 2003.

今井尚樹 , 金子晋丈 , 森川博之 , 青山友紀 ,
"3CEV 環境におけるパーソナルゲートウェイを用いたリソースハンドオフ機構 ,"
電子情報通信学会総合大会 , B-15-4, Sendai, Japan, Mar. 2003.
金子晋丈 , 市川雄一 , 中山雅哉 ,
"IPv4 および IPv6 に対応した時刻同期プロトコルの実装と評価 ,"
情報処理学会全国大会 , 4T9-1, Hachioji, Japan, Mar. 2003.
金子晋丈 , 小森田賢史 , 森川博之 , 青山友紀 ,
" セッション層モビリティサポートにおける高信頼データ転送の性能評価 ,"
電子情報通信学会ソサイエティ大会 , B-15-3, Niigata, Japan, Sep. 2003.
金子晋丈 , 林素娟 , 森川博之 , 青山友紀 ,
"Key-insulated 公開鍵を用いたセキュアなサービスモビリティの実現 ,"
情報処理学会研究報告 , 2003-UBI-2, pp.147-152, Kyoto, Japan, Nov. 2003.
金子晋丈 , 新藤晃浩 , 森川博之 , 青山友紀 ,
" セッション層アソシエーションによる通信路管理技術 ,"
電子情報通信学会総合大会 , B-6-43, Tokyo, Japan, Mar. 2004.
新藤晃浩 , 金子晋丈 , 森川博之 , 青山友紀 ,
" サービスモビリティを考慮したセッション管理機構 ,"
電子情報通信学会総合大会 , B-6-44, Tokyo, Japan, Mar. 2004.
小森田賢史 , 金子晋丈 , 森川博之 , 青山友紀 ,
"CommoNet: 基地局間連携による自律分散的マイクロモビリティサポート ,"
電子情報通信学会総合大会 , B-6-65, Tokyo, Japan, Mar. 2004.

吉田宣和 , 金子晋丈 , 王溪 , 森川博之 , 青山友紀 ,
" カットスルーパスを用いた適応的 IP ルーティングの実現 ,"
電子情報通信学会総合大会 , B-6-165, Tokyo, Japan, Mar. 2004.
新藤晃浩 , 金子晋丈 , 森川博之 , 青山友紀 ,
" サービスモビリティを考慮したセッション管理機構の設計と実装 ,"
電子情報通信学会技術研究報告 , IN2003-260, NS2003-305, Okinawa, Japan, Mar. 2004.
吉田宣和 , 金子晋丈 , 王溪 , 森川博之 , 青山友紀 ,
" ドメイン間パスを用いたショートカット IP ルーティング機構 ,
" 電子情報通信学会技術研究報告 , IN2003-309, Okinawa, Japan, Mar. 2004.
ダムラクス タナプーム , グエン ホアイソン , 金子晋丈 , 森川博之 , 青山友紀 ,
"IEEE 802.11 省電力端末の TCP プロキシ搭載 AP 間ハンドオフ機能 ,"
情報処理学会研究報告 , MBL-28-03, Tokyo, Japan, Mar. 2004.
情報処理学会 モバイルコンピューティングとユビキタス通信研究会 (MBL) 優秀論文 (論文誌推薦)
金子晋丈 , 森川博之 , 青山友紀 ,
" 通信路管理を実現するセッション層アソシエーション ,"
電子情報通信学会技術研究報告 , MoMuC2004-20, Okinawa, Japan, May. 2004.
小森田賢史 , 金子晋丈 , 森川博之 , 青山友紀 ,
" 自律分散的マイクロモビリティサポートのための基地局間マルチホップ網 ,"
電子情報通信学会技術研究報告 , MoMuC2004-26, Okinawa, Japan, May 2004.
金子晋丈 , 森川博之 , 青山友紀 ,
" サービスモビリティから見た通信の抽象化 ,"
電子情報通信学会技術研究報告 , MoMuC2004-36, Kyoto, Japan, Jul. 2004.
金子晋丈 , 森川博之 , 青山友紀 ,
" 人を主体とした安全で柔軟な通信サービスを実現する通信の抽象化 ,"
電子情報通信学会ソサイエティ大会 , B-6-42, Tokushima, Japan, Sep. 2004.
金子晋丈 , 栗田弘之 , 斉藤哲也 , 森川博之 , 青山友紀 ,
" セッション層アーキテクチャを用いた遠隔会議システムの設計と実装 ,"
電子情報通信学会総合大会 , B-6-4, Osaka, Japan, Mar. 2005.
斉藤哲也 , 金子晋丈 , 森川博之 , 青山友紀 , 羽鳥光俊 ,
" セッション層アーキテクチャにおける実通信機構の設計と実装 ,"
電子情報通信学会総合大会 , B-6-5, Osaka, Japan, Mar. 2005.
栗田弘之 , 金子晋丈 , 森川博之 , 青山友紀 ,
" セッション層アーキテクチャにおけるセキュリティゲートウェイの実現 ,"
電子情報通信学会総合大会 , B-6-6, Osaka, Japan, Mar. 2005.
丸山達也 , 松本延孝 , 金子晋丈 , 森川博之 , 青山友紀 ,
" 光パケットルーティングのためのブロードキャストを用いた経路情報広告手法 ,"
電子情報通信学会総合大会 , B-12-19, Osaka, Japan, Mar. 2005.
松本延孝 , 丸山達也 , 金子晋丈 , 森川博之 , 青山友紀 ,
" 中継ルータのアドレス部分処理による光パケットルーティング ,"
電子情報通信学会総合大会 , B-12-20, Osaka, Japan, Mar. 2005.

後郷和孝 , 渋井理恵 , 神谷弘樹 , 玉載旭 , 小森田賢 , 藤巻聡美 , 金子晋丈 , 寺岡文男 ,
" リンク層情報を利用したネットワーク層主導高速ハンドオーバ機構の設計と実装 ,"
電子情報通信学会技術研究報告 , MoMuC2005-3, Tokyo, Japan, May. 2005.
栗田弘之 , 金子晋丈 , 森川博之 , 青山友紀 ,
" セッション層アーキテクチャにおけるフロー情報を用いた通信資源管理機構 ,"
電子情報通信学会技術研究報告 , MoMuC2005-6, Tokyo, Japan, May. 2005.
丸山達也 , 松本延孝 , 金子晋丈 , 今泉英明 , 森川博之 , 青山友紀 ,
" 光パケットのソースルーティングに適した経路情報交換手法 ,"
電子情報通信学会技術研究報告 , PN2005-28, Obihiro, Japan, Aug. 2005.
金子晋丈 , 森川博之 , 青山友紀 ,
" セッション層アーキテクチャにおける複数フローの連携動作機構の設計 ,"
電子情報通信学会ソサイエティ大会 , B-6-27, Sapporo, Japan, Sep. 2005.
斉藤哲也 , 金子晋丈 , 森川博之 , 青山友紀 , 羽鳥光俊 ,
" セッション層アーキテクチャを用いた WWW 閲覧システムの設計と実装 ,"
電子情報通信学会総合大会 , B-6-52, Sapporo, Japan, Sep. 2005.
丸山達也 , 松本延孝 , 金子晋丈 , 今泉英明 , 森川博之 , 青山友紀 ,
" 光パケットルーティングにおけるトリガ更新型経路情報交換手法 ,"
電子情報通信学会総合大会 , B-12-28, Sapporo, Japan, Sep. 2005.
松本延孝 , 丸山達也 , 金子晋丈 , 今泉英明 , 森川博之 , 青山友紀 ,
" 簡略な転送処理による光パケットルーティング ,"
電子情報通信学会技術研究報告 , PN2005-43, Yokohama, Japan, Oct. 2005.

謝辞

まず，本研究にあたり終始御指導を賜った青山友紀教授，森川博之助教授に深甚なる謝意を表します．青山教授，森川助教授には，研究内容に関する有益な御助言以外にも，生活面においても種々の御教示を頂きました．

また，研究室において数々の便宜を図ってくださった渡邊廣次助手をはじめとする職員の方々，秘書の皆様に深く感謝いたします．

本研究で示したセッション層構成は，多くの方々の御尽力と御協力のおかげをもってここに一つの形を提示するに至りました．セッション層モビリティサポートの研究を始めたときから温かく見守って下さっている今井尚樹さん，セッション層モビリティサポートを手探りで追い求めていた時期に最初の実装を行って頂いた河内祐介さん，河内さんの実装を拡張してくれた鹿島拓也君，一つの形が見えてきた時期に二度目のセッション層モビリティサポートの実装を行ってくれた小森田賢史君，セッション層モビリティサポートからセッション層アーキテクチャへと舵を切るきっかけを生む議論と小森田君の実装をもとにプロトタイプを作成してくれた新藤晃浩君，現在のセッション層の実装を手伝ってくれた栗田弘之君と斉藤哲也君には，ここに深く感謝の意を表します．そして，セッション層に関して数え切れないほどの議論をさせて頂きました諸先生方，先輩方，同輩，後輩の方々に深く感謝致します．

本研究は筆者個人の力でできたものではなく，学内外を問わず多数の方々との議論，御教示や御協力を頂いてなし得たものです．最後に，これらの方々と私を陰ながら支えてくれてた家族に心から感謝の意を表します．

2005 年 12 月 15 日