社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－MBL－36（49）
2006－UBI－10（49）
2006／2／17

TranSwitch:ネットワークフロー毎における最適な TCP への動的切替機構

山下 勝司[1]　高橋 ひとみ[2]　斉藤 匡人[2]　徳田 英幸[2,3]

[1] 慶應義塾大学 総合政策学部 [2] 慶應義塾大学大学院 政策・メディア研究科 [3] 慶應義塾大学 環境情報学部

近年, 無線ネットワーク用 TCP や広帯域ネットワーク用 TCP といった特定のネットワークの性質を考慮して設計された新規 TCP が数多く提案されている. これらの新規 TCP は想定されたネットワーク環境で汎用性を重視した TCP よりも高い性能を発揮する. しかし, 現在一般に広く普及している OS は TCP を一つしか導入できないため, 多様なネットワーク環境に対応できる汎用性の高い TCP を用いる必要がある. このため, 特定のネットワークでの利用を想定した新規 TCP を現在の OS で利用することは難しい. 本論文ではこの問題を解決するため,OS へ複数の TCP を導入可能にし, ネットワークフロー毎に TCP を選択可能とする機構として TranSwitch を提案する. 本論文ではこの TranSwitch の設計と実装について述べ, 評価結果を用いて TranSwitch の有効性を示す.

TranSwitch : an Architecture for Dynamically Switching to Suitable TCP for Each Network Flow

Katsushi Yamashita[1]　**Hitomi Takahashi**[2]　**Masato Saito**[2]　**Hideyuki Tokuda**[2,3]

[1]Faculty of Policy Management, Keio University

[2]Graduate School of Media and Governance, Keio University

[3]Faculty of Environmental Information, Keio University

A lot of improved TCP have been proposed for correspondance to diversification of the network, such as TCPs for long fat pipe networks and those for wireless networks. These TCPs achieve higher performance than general TCPs in the assumed network environment. The TCPs are, however, not deployed widely due to some reasons. One of the reasons is that for current OS's do not support multiple different TCP, it is necessary that the only TCP can correspond to various network environments. We propose an architecture called TranSwitch to solve this problem. TranSwitch enables end users to introduce two or more TCPs into an OS, and switch a TCP according to a network flow to another. In this paper, we present a design and implementation of TranSwitch and effectiveness of TranSwitch with results of evaluation.

1. 背　　景

近年, 通信技術の発達により, 通信速度, 通信形態においてコンピュータネットワークが多様化している. インターネットサービスプロバイダが持つコアネットワークの通信速度は数 10Gbps と広帯域化し, 一般用のインターネット接続においても, ADSL や FTTH といったブロードバンド接続の通信速度が数 Mbps から 100 Mbps と広帯域化している. 同様に LAN (Local Area Network) においても 100 Mbps から 10 Gbps と高い通信速度が実現されている. 一方で, ISDN や携帯電話等のモバイル機器をモデムとして用いた低速なインターネット接続も広く利用されている. これらの低速なインターネット接続の通信速度は数 10 Kbps から数 100 Kbps であり, 前述の高速化したネットワークと通信速度の面において大きな開きが生じている. 通信形態においては, IEEE 802.3[1] を用いた有線通信や IEEE 802.11[2] を用いた無線通信等のネットワークが混在している.

このようなネットワークの多様化に対応する形で, それぞれのネットワークの特性を考慮した新規 TCP がこれまでに数多く提案されている. これらの TCP の代表的な例として, 無線ネットワーク用の TCP[3)4)5)], 広帯域高遅延ネットワーク (LFN:Long Fat Network) 用の TCP[6)7)8)9)] などがある. 汎用的な TCP はどのような通信環境においても比較的高いパフォーマンスを発揮し, 特定の通信環境でパフォーマンスが著しく低下することはない. 一方, これらの新規 TCP はそれぞれの想定された通信環境において, 汎用的な TCP より高いパフォーマンスを発揮するが, ネットワーク環境によっては汎用的な TCP より低いパフォーマンスしか発揮できないという特徴がある.

2. 問 題 意 識

無線ネットワークや広帯域高遅延ネットワーク等の特定環境において, 汎用的な TCP より高いパフォーマンスを発揮する新規 TCP がこれまで多く提案されている. しかし, 現在これらの新規 TCP は一般に広く普及していない. その原因の一つが現状一般に広く普及する OS の TCP の利用形態にある. OS はトランスポート層プロトコルとして, TCP, UDP, SCTP[10)] といった複数のプロトコルを所持できるが, TCP としては一つしか所持できない. 一つの TCP で多様なネットワーク環境に対応する必要性から, 汎用性を重視した TCP を OS は利用する必要がある.

LAN や専用回線を用いた通信のようにネットワークの帯域や遅延が特定できる場合や, 無線通信のみを利用するといった通信形態が特定できる場合, そのネットワーク環境に特化した TCP の利用が可能である. しかしインター

ネットを用いた通信の場合, 通信相手が多様であり, ネットワークの輻輳状況も常に変化するため, 遅延, 帯域が多様化する. また, ノートパソコンの利用などにより有線通信と無線通信が併用されることも多い. このように利用するネットワーク環境が特定できない場合, 多様なネットワーク環境への柔軟な対応が TCP には求められる. そのため, OS は汎用性を重視した TCP を利用する必要があり, 特定環境に特化した TCP の利用は難しい.

この問題点を解決するための機構として本研究は TranSwitch を提案する. TranSwitch は OS に複数の TCP を導入し, これら複数の TCP からアプリケーションが利用する TCP をネットワークフロー毎に選択可能にする. OS へ複数の TCP を導入し, ネットワークフロー毎に異なる TCP を利用することで, 汎用性を重視した TCP のみを利用する必要がなくなり, それぞれのネットワークフローの特性に適した TCP が利用可能となる.

3. 設　　計

3.1 概　　要

本論文では TranSwitch を提案する. TranSwitch は OS に複数の TCP を導入し, これら複数の TCP からアプリケーションが利用する TCP をネットワークフロー毎に選択可能にする. また, エンドユーザによるユーザレイヤからの TCP の選択により, TranSwitch は通信環境の多様性や変化に柔軟かつ即時的な対応が可能である.

3.2 想定シナリオ

TranSwitch を用いて OS に複数の TCP を導入することで, 汎用性を重視した TCP のみの利用が必要でなくなり, それぞれのネットワークフローの特性に適した TCP の利用が可能となる. これを図 1 に示す. 図 1 では左に通常の OS を用いた場合, 右に TranSwitch を用いた場合の TCP の利用を表している. 通常の OS は TCP を一つしか所持できないため, 全てのネットワークフローで汎用性の高い TCP を利用している. 一方, 右の TranSwitch を用いた場合では,OS は複数の TCP を所持できるため, 汎用性の高い TCP だけでなく広帯域ネットワーク用 TCP, 無線ネットワーク用 TCP を所持している. ネットワークフロー A の通信は通信ホストとの広帯域な通信が可能であるため, 広帯域ネットワーク用 TCP を利用し, ネットワークフロー B の通信は無線を用いた通信を行うため, 無線ネットワーク用 TCP を利用し, ネットワークフロー C の通信はネットワークの特性が不明なため, 従来と同様に汎用性の高い TCP を利用している. このように TranSwitch を用いてより適した TCP を各ネットワークフローで利用することにより, すべてのネットワークフローで汎用性を重視した TCP を利用する場合と比較し, より効率の良い通信が可能である.

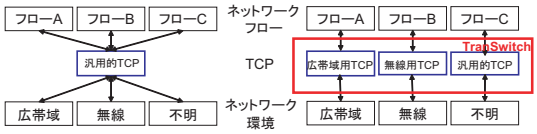

図 1　ネットワークフロー毎の異なる TCP の利用

3.3 機　　能

TranSwitch は OS に複数の TCP を導入可能にし, エンドユーザはこれらの TCP のネットワークフロー毎の利用が可能となる. また, この選択はユーザレイヤからエンドユーザが設定可能である. よって, TranSwitch は以下の機能を持つ.

- 複数 TCP 混在機能
  TranSwitch は TCP を一つしか導入できない従来の OS を複数の TCP を所持可能に拡張する. TranSwitch へ追加する TCP の実装形式はエンドユーザによる簡易な追加および削除を実現するためカーネルモジュールを用いる. 追加 TCP は TranSwitch 上での利用を可能にするため, TranSwitch が定める TranSwitch 用 TCP フォーマットに準拠して実装される必要がある.

- 定義情報設定ユーザインタフェース
  通信形態, 遅延, 帯域といったネットワークの特性はネットワークフロー毎に異なる. そのため, 特定のネットワークの特性に最適化された TCP を適切に選択するためには, ネットワークフロー単位での TCP 選択が必要である. よって, TranSwitch では TCP の選択をネットワークフロー単位で行い, このネットワークフローと TCP を関連付けた定義情報を TranSwitch はカーネルレイヤで保持する. また, ネットワークの特性やアプリケーション特性は動的に変化するため, 適した TCP の選択には即時的かつ柔軟に TCP を選択できる必要がある. このため TranSwitch ではネットワークフローと TCP の関連付けをユーザインタフェースを用いてエンドユーザが設定する.

- TCP 切替機能
  TranSwitch はネットワークフロー毎に TCP を切替る. TranSwitch に追加される TCP は TCP の処理全てではなく, カーネル内で実装された TCP との差分のみを関数単位で用意する. 追加 TCP への処理の切替は, 本来の処理される関数から差分として実装された関数への処理の遷移によって実現される. この関数を置換関数と定義し, 置換関数によって置換されるカーネル内の関数を被置換関数と定義する. TCP の切替処理は図 2 に示すように以下の手順で行われる.

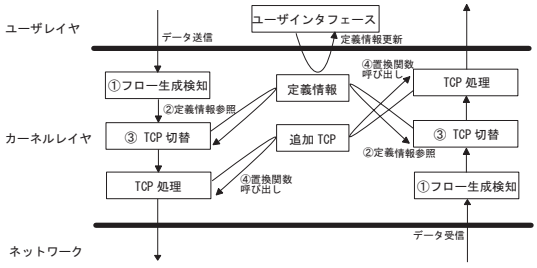

図 2　TranSwitch における TCP 切替処理

(1)　TCP を用いるネットワークフローの生成を検知

(2) 検知したネットワークフローと定義情報の照会
(3) 該当ソケットと TCP の関連付け
(4) TCP の処理で被置換関数に処理が遷移する際, 置換関数へ処理を切替

4. 実　　装

今回の TranSwitch のプロトタイプ実装では Linux カーネル 2.4.32 と Fedora Core 1 を用いて実装を行った.

4.1 TCP 混在環境

4.1.1 TCP の追加および削除

TranSwitch は追加 TCP の実装形式としてカーネルモジュールを採用する. 従来の新規 TCP の配布形態はカーネルパッチを用いてカーネルのソースを上書きする方法が一般的であった. カーネルモジュールを利用することで, 従来のカーネルパッチの利用と比較し, カーネルコンパイルが不要なため即時的な TCP の追加, 削除が可能であり, エンドユーザによる容易な TCP の追加および削除が可能である. このようなカーネルモジュール化した追加 TCP を TranSwitch 上で利用可能にするため, 追加 TCP は次節で述べる TranSwitch が定める TranSwitch 用 TCP フォーマットに準拠する必要がある.

4.1.2 TranSwitch 用 TCP フォーマット

TranSwitch に追加する TCP は以下の TranSwitch 用 TCP フォーマット全てに準拠して実装される必要がある. 図 3 に TranSwitch 用 TCP フォーマットに準拠して定義された TCP の例を示す.

```
void sample_tcp_select_window(void){
        …
}

static struct trsw sample_tcp = {
        .tcp_select_window = sample_tcp_select_window,
        .name  ="sample_tcp"
};

int init_module(void){
        trsw_regist(&sample_tcp);
        return 0;
}

void cleanup_module(void){
        trsw_unregist(&sample_tcp);
}
```

図 3　TranSwitch 用 TCP の実装例

- TranSwitch への登録関数および削除関数の実装
  図 3 に表すように trsw_regist 関数を用いて trsw 構造体を TranSwitch に登録することで TranSwitch への追加が可能である. 同様に trsw 構造体を引数として trsw_unregist 関数を用いることで TranSwitch からの削除が可能である. 追加 TCP への処理の遷移等はこの trsw 構造体を経由して行われる.

- TCP 名の表記
  TranSwitch はエンドユーザが定義した定義情報に従って TCP を切替る. この際, TCP を特定するために一意な TCP の識別子が必要となる. そのため TCP はカーネルモジュール内でこれを明記する必要がある. 図 3 では trsw 構造体の name 変数に sample_TCP として代入することで TranSwitch に sample_TCP として識別される.

- 置換対応の表記
  TranSwitch による TCP の切替は被置換関数から置換関数への処理の遷移によって実現する. その際, カーネルモジュール内に実装されたどの置換関数がカーネル内で実装されたどの被置換関数に対応するか表記する必要がある. 図 3 では trsw 構造体内で関数ポインタを用いて tcp_select_window 関数から sample_tcp_select_window 関数への置換を表現している.

- 置換関数の実装
  追加 TCP は置換対応において置換関数として表記した関数をカーネルモジュール内で実装する必要がある. 図 3 では置換関数として登録した sample_tcp_select_window 関数の定義を行っている. sample_TCP を利用すると定義されたネットワークフローは tcp_select_window 関数に処理が移った際, trsw 構造体の置換対応を参照して sample_tcp_select_window 関数を呼び出す.

4.2 定義情報設定ユーザインタフェース

TranSwitch はネットワークフロー毎に TCP を切替る. そのため, 定義情報はネットワークフローを表す送受信ホストの IP アドレス, ポート番号と利用する TCP の名前によって構成される. 即時的かつ柔軟な TCP の選択を可能にするため, TranSwitch ではユーザランドからエンドユーザがコマンドを用いた定義情報の追加, 削除が可能である. これを図 4 に示す. 図 4 における例 1 は受信ホストの IP

```
例1.
% trswadd -i 10 -si 192.168.1.5 -di 192.168.1.3 -sp 20 -dp 100-200 fast

例2.
% trswadd -si 127.0.0.1 westwood

例3.
% trswdel 10

例4.
% trswlist
id      src IP        dst IP     src port     dst port       TCP
1       127.0.0.1     0.0.0.0       0            0         westwood
```

図 4　定義情報の入力例

アドレスが 192.168.1.3, 受信ホストのポート番号が 100 から 200, 送信ホストの IP アドレスが 192.168.1.5, 送信ホストのポート番号が 20 である全てのネットワークフローの TCP として FAST[8] を利用し, この定義の ID を 10 とする設定の例である. 例 2 は送信ホストの IP アドレスを 127.0.0.1 とする全てのネットワークフローの TCP として Westwood[3] を利用するという設定の例である. 例 2 の場合, 送信ホストの IP アドレス, 送信ホストおよび受信ホストのポート番号, 定義情報の ID が入力されていない. ネットワークフローを示す値で特に定義されていない項目は全ての値に対応することを意味する. また, 定義情報の ID は

入力されていない場合, 既存の定義情報に利用されてない値の最小値が自動で割り当てられる. そして例 3 では, 事前に追加されている定義を trswdel コマンドで ID を指定して削除している. 最後の例 4 では, trswlist コマンドを用いてこれまで定義された定義情報のリストを一覧表示している.

4.3 TCP 切替機能

生成を検知したネットワークフローと合致するネットワークフローが定義情報にある場合, 選択された TCP と生成されたネットワークフローを関連付けるため, そのネットワークフローのソケットに TCP を登録する. これを図 5 に示す. 図 5 ではソケットそれぞれに TCP が関連付けられていることを示している. ソケット A には汎用性の高い TCP である Reno が, ソケット B には追加 TCP である Sample TCP がそれぞれ関連付けられている. tcp_transmit_skb 関数から tcp_select_window 関数を呼び出す際, ソケットが持つ TCP への参照に従って次の関数を呼び出している. ソケット A は Reno への参照をもつため, Reno を実装しているカーネルの tcp_select_window 関数を呼び出す. 一方, ソケット B は TCP への参照として Sample TCP への参照を持つため, カーネルモジュール内にある sample_tcp_select_window 関数を呼び出す. TranSwitch はソケットが持つ TCP への参照を利用することで, カーネルモジュール内で実装された新規 TCP の関数群へ適切な処理の切替を行う.

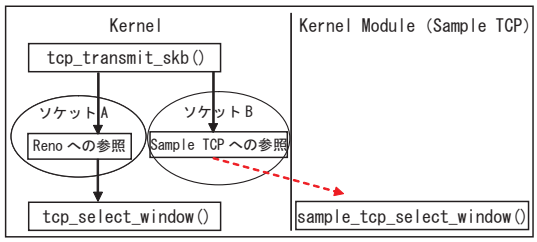

図 5 ソケットと TCP の関連付け

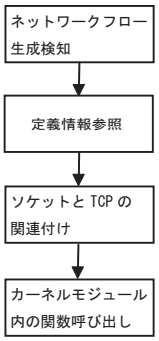

図 6 TranSwitch による TCP 切替までの動作

TCP の切替における一連の処理を図 6 に示す. まず, TranSwitch は TCP を用いるネットワークフローの生成を検知し, 定義情報を参照する. 生成を検知したネットワークフローと合致するネットワークフローが定義情報にある場合, その定義で関連付けられている TCP をそのネットワークフローの TCP としてソケットに関連付ける. そして TCP の処理中で置換関数として登録された関数の処理を行う際は, ソケットに登録された TCP を実装したカーネルモジュール内の関数に処理を移す. 以上の一連の処理により, TranSwitch は TCP の切替を実現する.

5. 評　　価

TranSwitch はネットワークフロー毎に利用する TCP の選択が可能である. この機能は前述したように, TranSwitch が関数を置換することで実現される. この際, 置換関数へ処理を切替えることでオーバヘッドが生じる. 本評価では TranSwitch のオーバーヘッドとして, この関数の置換によって生じるスループットへの影響を評価した.

5.1 評 価 方 法

本評価では表 2 に示す評価機器 2 台間の TCP スループットを iperf 2.0.2 を用いて計測した. スループットを計測する際, 評価機器のハードウェア性能がボトルネックとなり, TranSwitch によるオーバヘッドがそのボトルネックに吸収される場合, オーバヘッドの正確な計測ができない. これを防ぐため, 本評価では図 7 に示すように Dummynet を用いて 2 台のホスト間の遅延をエミュレートしたネットワークを評価環境として利用する. TCP のスループットは帯域遅延積によって決定されるため, 遅延の上昇によりスループットは減少する. 本評価では遅延を 10 ms に設定することで, スループットをボトルネックの影響を受けない値まで下げる. これにより, TranSwitch によるオーバヘッドが評価機器のハードウェア性能によるボトルネックに吸収されることを防ぐ.

表 1 評価機器

項目	説明
PC	IBM Thinkpad t42p
OS	Fedora Core 1
カーネル	Linux 2.4.32
CPU	Intel Pentium M 2GHz
NIC	1 Gigabit Ethernet
RAM	PC2700 2GByte

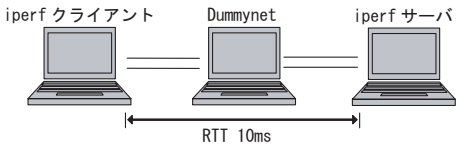

図 7 評価用ネットワーク

TranSwitch のオーバヘッド計測の際, TCP のアルゴリズムの違いによるスループットの差が生じることを防ぐため, 本評価の評価端末が当初からカーネル内で実装している TCP である Reno を TranSwitch が切替る TCP として利用した. よって, TranSwitch による TCP の切替はカーネル実装の Reno から TranSwitch における実装形式の Reno へと行われる. 本評価では TranSwitch の全ての置

換関数をカーネル実装と全く同じ実装でカーネルモジュール内で実装した．この TCP 切替のスループットを計測することで，TranSwitch による TCP の切替処理のオーバヘッドが計測可能である．

5.2 評価結果

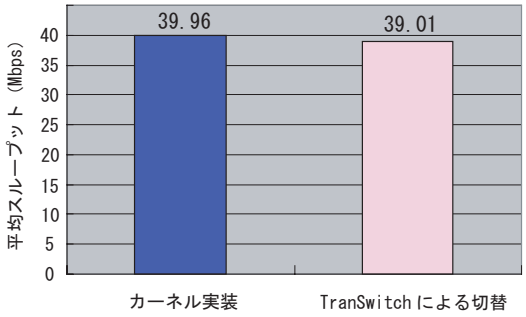

図 8 TranSwitch の利用によるオーバヘッド

図 8 に評価結果を示す．カーネル実装による Reno の平均スループットが 39.96 Mbps，カーネルモジュールとして実装され，TranSwitch によって呼び出される Reno の平均スループットが 39.01 Mbps である．なお，標準偏差はカーネル実装による Reno が 0.08 であり，TranSwitch によって呼び出される Reno が 0.07 であった．平均スループットの値の差は 0.85 Mbps であり，TranSwitch の切替によりスループットは 2.1 % 減少している．このように TranSwitch の利用によりオーバヘッドが発生し，スループットが 2.1 %減少するが，新規 TCP の多くが今回計測したオーバヘッド以上の改善を見せる．本評価により，TranSwitch の TCP 切替によってオーバヘッドは生じるものの，TCP の切替によるスループットの向上において十分に有効であると示すことができた．

6. 関連研究

6.1 WAD

ソケット通信ではソケットが持つバッファサイズを超える容量の送受信はできない．そのため高いスループットで通信を行う場合，ソケットの持つバッファサイズがボトルネックとなり，ウインドウサイズが増加しなくなる問題がある．この問題はソケットバッファサイズをより大きな値に設定することにより解決できる．WAD (Work Around Daemon)[11] はネットワークフロー毎にソケットバッファサイズ等のチューニングが可能な機構である．WAD を用いることにより，高いスループットで通信を行う必要のあるネットワークフローのソケットバッファサイズを増加させ，バッファサイズがボトルネックとなりウインドウサイズが向上しない問題を改善できる．このように WAD はネットワークフロー毎のソケットバッファ等のチューニングが可能であるが，TranSwitch のような TCP の切替は不可能である．また，WAD はネットワークフローとチューニングを関連付けた定義情報をユーザランドで動作するデーモンで管理し，TCP のフローが生成される毎にこの定義情報を参照する．そのためネットワークフローが生成される際，一度カーネルレイヤからユーザレイヤに処理を移し，定義情報を検索する必要があり，この処理がオーバヘッドとなる．

6.2 X-Kernel

X-Kernel[12] は各プロトコルをモジュールとして管理し，これらを組み合わせることでプロトコルスタックが再編可能なマイクロカーネルである．これにより新規プロトコルをプロトコルスタックに組み込むことが可能となる．しかし，X-Kernel はマイクロカーネルとして動作し，プロトコルをユーザランドで管理するため，導入により，ネットワークのアーキテクチャが大きく異なる．一方で TranSwitch は現在普及している OS の TCP/IP 実装において最小限の変更で TCP の切替を可能にする．このため TranSwitch 導入後も従来の OS のネットワークアーキテクチャに依存して動作する他の機構が利用可能であり，この点において X-Kernel より優れている．

6.3 AS-TCP

AS-TCP (Application Specific TCP)[13] は TranSwitch と同様に OS に複数の TCP を導入し，アプリケーションが利用する TCP をエンドユーザが選択可能な機構である．AS-TCP における TCP の選択はアプリケーション毎に行い，TranSwitch のネットワークフロー毎の TCP の選択と異なる．同一のアプリケーションによる通信であっても通信ホストが異なる場合，通信経路が異なるため，輻輳，遅延，ボトルネックリンクといったネットワークの特性が異なる．そのためアプリケーションと TCP の関連付けではアプリケーションの通信の特性を考慮した TCP の選択は可能であるが，それぞれのネットワークフローの特性を考慮した TCP の選択は行えないという問題点がある．TranSwitch はネットワークフロー毎の TCP の選択が可能なため，アプリケーションの通信の特性だけでなく，ネットワークフローの特性を考慮した TCP の選択が可能という点において AS-TCP より優れている．

7. まとめと今後の課題

7.1 ま と め

本論文ではネットワークフロー毎により最適な TCP を利用可能とする機構として TranSwitch を提案した．昨今のネットワーク環境は通信速度や通信形態の面において多様化しており，これを背景として，広帯域ネットワーク用 TCP や無線ネットワーク用 TCP などの特定のネットワーク環境に最適化した TCP がこれまで多く提案されている．しかし，従来の OS は TCP を一つしか所持できず，また利用するネットワーク環境が特定できないことが多いため，汎用性の高い TCP を利用する必要があり，これら特定環境での利用を想定した TCP を従来の OS で利用するのは難しい．この問題点の解決策として，複数の TCP を所持可能に OS を拡張し，ネットワークフロー毎に利用する TCP をエンドユーザが選択可能にする機構として TranSwitch を提案した．本論文ではこの TranSwitch の設計・実装について述べ，プロトタイプ実装の評価結果を用いて TranSwitch の有効性を示した．

7.2 今後の課題

TranSwitch の今後の課題として以下の三点が考えられる.

- 通信途中における TCP の動的切替
  現在の TranSwitch の設計ではネットワークフローが生成される際, エンドユーザによって予め定義されている TCP がネットワークフローに関連付けられる. ネットワークフローの生成以降はネットワークフローと TCP の関連付けは変更されない. これは通信最中に TCP を変更した場合, TCP の処理の一貫性を失う恐れがあるためである. しかし, ネットワークの特性やアプリケーションの特性などの TCP の選択要因の全てを通信が開始される前の段階で把握できるとは限らず, 通信が開始された後により適した TCP の存在に気づく場合がある. また, インターネットのような公衆回線の輻輳や遅延は通信の最中で変化するため, 当初適切として選択した TCP が通信の最中で不適切な TCP となる可能性もある. これらの問題は通信の最中にネットワークフローが利用する TCP を変更することで解決できる. このような通信途中における TCP の動的切替を実現することで TranSwitch の有効性を更に向上できる.

- 定義情報の定義における利便性向上
  今回の TranSwitch の実装ではネットワークフローと TCP の関連付けをネットワークフローを表現する送受信ホストの IP アドレス, ポート番号と TCP 識別子で行っている. しかし, これ以外の定義方法が豊富であるほうが定義を行いやすい. 例えば, 無線ネットワーク用 TCP を無線通信を行うすべてのネットワークフローで利用する場合, IP アドレスやポート番号ではなく無線 NIC のネットワークインタフェース名と TCP 識別子を用いてネットワークフローと TCP の関連付けを行う方が容易である. このようにより多くの定義方法を用いた定義情報の定義を可能にすることで, より容易な TranSwitch の利用が可能となる.

- TCP 選択補助機構
  TranSwitch では TCP とネットワークフローの関連付けの定義をエンドユーザが行う. これは TCP を選択する上で要因となる輻輳, 遅延, 通信形態といったネットワーク環境やアプリケーションの特性等の情報を最も詳細に把握可能であるのがエンドユーザだからである. しかし, ネットワークに関する知識の欠如などによりエンドユーザが不適切な TCP 選択を行う可能性がある. この場合のスループットは TranSwitch を利用せずに汎用性を重視した TCP を利用した場合に得られるスループットよりも低くなる恐れがある. この問題は TranSwitch がエンドユーザに代わって TCP の選択を行うなど, TranSwitch がエンドユーザの TCP の選択を補助する機能を持つことで解決できる. このような TCP 選択補助機構により TranSwitch の有効性を向上できる.

謝　　辞

本研究は「慶應義塾大学：デジタルメディア・コンテンツ統合機構」の一部として行われました。

参 考 文 献

1) IEEE Standards for Local Area Networks: *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, 1985.
2) IEEE 802.11 Standard (IEEE Computer Society LAN WAN Standards Committee). *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.
3) S. Morris and C. Casetti. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *MobiCom 2001: The Seventh Annual International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.
4) K. Brown and S. Singh. M-TCP: TCP for mobile cellular networks. ACM Comp. Comm. Review, vol. 27, 1997.
5) Zygmunt J. Haas. Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems. 3rd InternationalWorkshop on Mobile Multimedia Communications, September 1995.
6) S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.
7) T. Kelly. Scalable tcp: Improving performance in highspeed wide area networks. In *Proceedings of the First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*, Feb. 2003.
8) C. Jin, D. X. Wei, and S. H. Low. FAST TCP: motivation, architecture, algorithms, performance. *IEEE Infocom*, March 2004
9) L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control for Fast, Long Distance Networks. *IEEE INFOCOM*, March 2004.
10) R. Stewart, Q. Xie, et al. Stream Control Transmission Protocol. RFC 2960, October 2000.
11) T. Dunigan, M. Mathis, and B. Tierney. A TCP tuning daemon. In *Proceedings of SuperComputing: High-Performance Networking and Computing*, Nov. 2002.
12) N. C. Hutchinson and L. L. Peterson. The x-kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, Jan. 1991.
13) 小野 祐介, 斉藤 俊介, 田中 康之, 寺岡 文男. 通信の特性に応じた TCP アルゴリズム切り替え機構の設計と実装. マルチメディア, 分散, 協調とモバイルシンポジウム (DICOMO2005), Jul. 2005.