ARMS: Application-level Concurrent Multipath Utilization on Reliable Communication

野澤 高弘[†] 本多 倫夫[‡] 榊原 寛[‡] 中澤 仁[§] 徳田 英幸[§]
Takahiro Nozawa Michio Honda Hiroshi Sakakibara Jin Nakazawa Hideyuki Tokuda
{takahiro,micchie,skk,jin,hxt}@ht.sfc.keio.ac.jp

概要

近年，複数のネットワークインタフェースを搭載したモバイル端末が普及している．しかし現状では，複数のネットワークインタフェースを持っていても同時に利用されていないため，ネットワークリソースが効率的に活用されていない．本研究では，複数のネットワークインタフェースを同時に活用し，異なるキャリアが提供するネットワークリソースを効率的に使うことによって通信効率を向上させる手法，ARMS を提案する．ARMS の特徴として，アプリケーション層で実現することが挙げられる．それにより既存のオペレーティングシステム及びネットワークプロトコルスタックに変更を加える必要がない．そのため，アプリケーション開発者自身によるマルチパス転送を実現可能である．ARMS ではトランスポートプロトコルに SCTP を用いることで，通信相手のアドレスや各パスの情報を取得し，また異なるパスを流れる複数のフロー間におけるデータの同期を行う．本研究では ARMS のプロトタイプ実装による評価を行い，単一インタフェースを利用した通信と比べて最大で 2.0 倍転送速度を向上させた．

1. はじめに

無線技術の発展と普及により，公共空間の多くで無線 LAN によるインターネットアクセスが提供されている．さらに，HSDPA や 802.16e などによる，より広範囲をカバーする無線 MAN によるインターネットアクセスも提供され始めている．様々な無線技術の実用化に伴い，複数の無線ネットワークインタフェースを搭載したモバイル端末が登場している．そのような端末は，複数の無線接続が可能な場所において複数の通信経路を利用可能であり，一般的にマルチホームと呼ばれる．図 1 にマルチホーム環境の例を示す．図 1 では，モバイルノードは Wireless WAN と Wireless LAN 両方の通信範囲に位置し，2 つのネットワークを使って通信可能である．

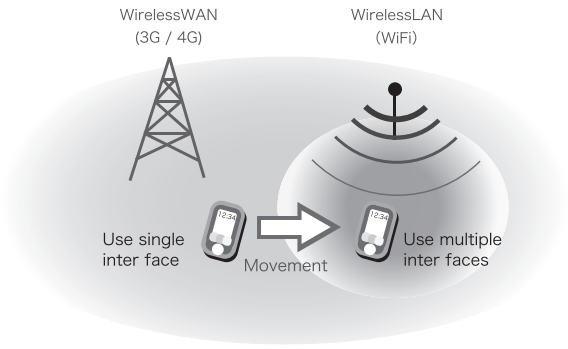

図 1: 想定環境

モバイル端末の高性能化に伴いそれらにおけるマルチメディアデータなど大容量データの転送が可能になり，ユーザはより多くのネットワーク資源を必要としている． しかし，現在のインターネットにおいては，アーキテクチャ及びプロトコル上の制約によりマルチホームのホストにおいても単一の NIC を利用した通信が一般的である．しかし，このような通信は資源活用の観点からは非効率的である．そのため，本研究では複数の NIC を用いて異なるキャリアによって提供されるネットワークを同時に利用することにより多くのネットワーク資源を確保し，効率的な通信を実現する手法である Application-level Concurrent Multipath Utilization on Reliable Communication(ARMS) を提案する．

ARMS の特徴として，トランスポートプロトコルに Stream Control Transmission Protocol (SCTP) [6] を用いることで各パスを流れる複数のフローの間でデータを同期し，信頼性のあるマルチパス転送を実現する．ARMS は現在のインターネット上におけるエンドノードに広く実装されたプロトコルアーキテクチャおよびオペレーティングシステムの変更を必要としないため，アプリケーション開発者が容易に利用できる．また，end-to-end の通信における送信側アプリケーションのみの実装で実現可能なことから，段階的に普及できる．

本論文は以下で構成される．2 章ではこれまでに提案されてきた複数経路を同時に利用する手法と問題点について述べる．3 章では ARMS の設計および実装について述べ，4 章では ARMS のプロトタイプ実装による実験結果について述べる．5 章で本論文をまとめ，今後の展望について述べる．

2. 関連研究

これまでに提案されてきた end-to-end で複数パスを利用して通信速度を向上させる手法として，本研究とは異なりオペレーティングシステム内で実現する手法が挙げられる．TCP を改良して実現している手法として pTCP [1]，mTCP [5]，AMS [10], R-M/TCP [9] が挙げ

[†]慶應義塾大学環境情報学部
[‡]慶應義塾大学政策・メディア研究科
[§]慶應義塾大学政策・メディア研究科，環境情報学部，JST-CREST

られる．SCTP を改良して実現している手法としては，CMT [2] が挙げられる．また，新たなトランスポートプロトコルにおいて実現している手法として，R- MTP [3] が挙げられる．これらの手法は，オペレーティングシステム内の実装を必要とし，ユーザ空間のみでは利用不可である．また，どの技術も標準化されておらず，CMT を除きどのオペレーティングシステムにも実装されていない．CMT は FreeBSD のみに実装されているが，その他のオペレーティングシステムでは実装されていない．また，これらの手法は，通常の単一経路を利用した TCP または SCTP のアプリケーションとの共存については触れていない．通常の TCP および SCTP のアプリケーションと複数パス同時利用を利用するアプリケーションが共存するためには，新たな API を実装し，各アプリケーションが複数パス同時利用を行うか行わないかを選択する必要がある．

3. ARMS Design

このような現状を踏まえ，本研究ではアプリケーションレベルでマルチパス転送を実現する手法である Application-level Concurrent Multipath Utilization on Reliable Communication(ARMS) を提案する．

ARMS はモバイル端末における通信を想定するため，動的に変化する異なる特性の通信経路を同時に利用した信頼性のある通信を実現する．ARMS の動作概要を図 2 に示す．ARMS では複数の宛先を有効に活用するために，アプリケーションにおいてパスの情報を取得し，それに基づいてデータを各宛先に転送する．以降では第一に，ARMS で使用するトランスポートプロトコルの選択について述べる．第二に ARMS の動作概要を述べ，第三に複数の宛先への最適な送信比率の決定方法を述べる．最後に複数の宛先があるときの宛先の指定方法を述べる．

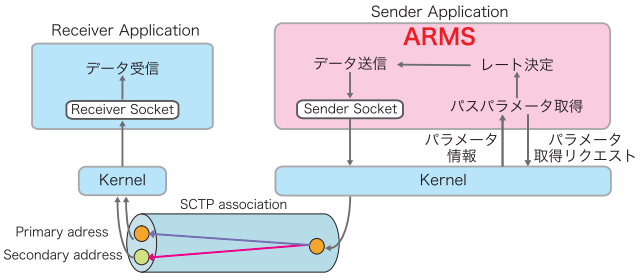

図 2: ARMS システム概要

3.1 トランスポートプロトコルの選択

アプリケーションにおいてマルチパス転送を行う際に重要なことの一つとして，トランスポートプロトコルの選択が挙げられる．本研究では信頼性のある通信を実現するため，TCP か SCTP が選択肢として考えられる．

- 通信相手のアドレスの取得
  複数パスを利用した通信を行うためには，まず通信相手がどのアドレスを持つのか検出する必要がある．アプリケーションにおいて通信相手のアドレスを取得するためには，アプリケーションが独自に通信相手とネゴシエーションすることによって通信相手の持つアドレスリストを取得する方法が考えられる．しかしこの手法では，アプリケーション開発者の負担が大きくなる問題がある．そのため，ARMS ではトランスポート層プロトコルである SCTP の機能を用いて通信相手の持つアドレスを取得する．

  SCTP は TCP と同様に信頼性のある通信を実現するトランスポートプロトコルであり，TCP の 3-Way Handshake にあたる 4-Way Handshake により end-to-end のコネクションを確立する． SCTP におけるコネクションはアソシエーションと呼ばれる．SCTP では，4-Way Handshake においてお互いが自身の利用可能なアドレスを通信相手に通知する．また，アプリケーションはアソシエーションにおける利用可能な宛先リストを SCTP が提供する API [7] によって取得できる．具体的には，*sctp_get_paddrs()* と呼ばれるシステムコールを用いる．そのため，アプリケーションは，アドレスリストのシグナリング機能を実装することなく通信相手のアドレスを取得できる．

- 動的なアドレスの変化
  本研究では移動体無線通信を前提として考えている．このような環境では端末の IP アドレスが動的に変化する．TCP は永久的に一組の IP アドレスの組を通信の単位として利用するため，端末の IP アドレスが変化した場合は新たなコネクションを再度生成する必要がある．一方で，古いコネクションと新たなコネクションの間でデータを同期することは困難である．SCTP は Dynamic Address Reconfiguration (ADDIP) [8] 機能によりアドレスが変化した際も同一のアソシエーション内で通信を維持できるため，端末の IP アドレスが変化した際も古いアドレスに対するデータ送信あるいは古いアドレスからのデータ送信と新たな IP アドレスを用いたデータ送信との間でデータが同期される．ARMS はデータ転送に SCTP を用いるため，端末の動的なアドレスの変化に適応可能である．

以上を踏まえ，ARMS ではトランスポートプロトコルに SCTP を用いる．SCTP は既に FreeBSD, Linux, Solaris に標準で搭載されている．また，FreeBSD の実装をベースにしたサードパーティーの実装として，Windows のドライバ実装，Mac OSX カーネルモジュール

の実装が存在する．

3.2 複数宛先へのデータ送信

本研究ではマルチパス転送を実現するために SCTP を用いるが，通常の SCTP のマルチホーム環境における動作ではマルチパス転送は実現できない．図 3 (a) に通常の SCTP のデータ転送と，図 3 (b) に ARMS によって行うデータ転送を示す．通常の SCTP のデータ転送ではプライマリパスにしかデータを転送せず，データが到達しなかった場合にのみセカンダリパスへデータが転送される．この転送方法では通信の接続性を維持する効果は期待できるが，ネットワーク資源を有効に使いきることはできず，通信速度の向上は見込めない．本研究では同時に利用可能なパスが複数存在したとき，図 3 (b) のように全てのパスを同時に用いる． それによりネットワーク資源を最大限に利用し，高速にデータ送信を行う．

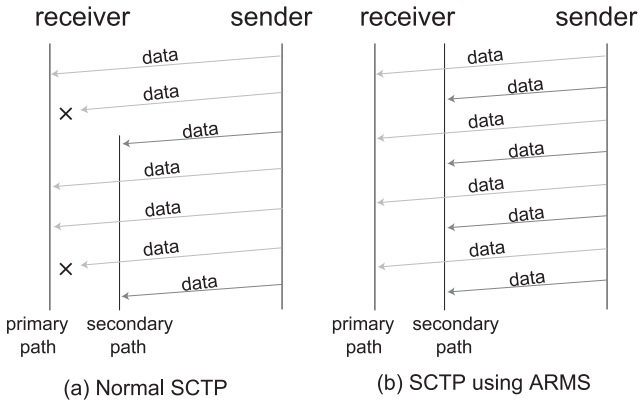

図 3: SCTP のマルチパス転送の比較

ARMS では図 3 (b) のように指定したアドレスへデータを転送するために，SCTP が提供する API である *sctp_sendmsg()* 関数 [7] を用いて実現する．*sctp_sendmsg()* では引数にて宛先アドレスを明示的に指定して送信することが可能である．その宛先がプライマリアドレスでない場合は，SCTP 内部で宛先をプライマリアドレスに変更されてしまうが．引数のパラメータに *SCTP_ADDR_OVER* フラグをたてる事によりプライマリアドレス以外のパスへデータを送信できる．そのため，ARMS ではデータ送信に *sctp_sendmsg()* をこのフラグと共に用いる．

3.3 送信比率決定

本研究は，複数の異なる種類のネットワーク環境に接続されている環境 (図 4) を前提としているが，この環境下においてパルチパスを実現するには以下の問題点が挙げられる．

- 帯域幅
  図 4 で示すマルチパス環境下では，Link 1 の帯域幅が 2Mbps，Link 2 の帯域幅が 8Mbps と異なる．このときパケットを各パスへ同比率で送信を行うと，Link 2 (帯域幅の大きいリンク) のパスを利用しきれない．よって，全てのパスの帯域を効率的に利用するために最適な転送レートの決定が必要となる．

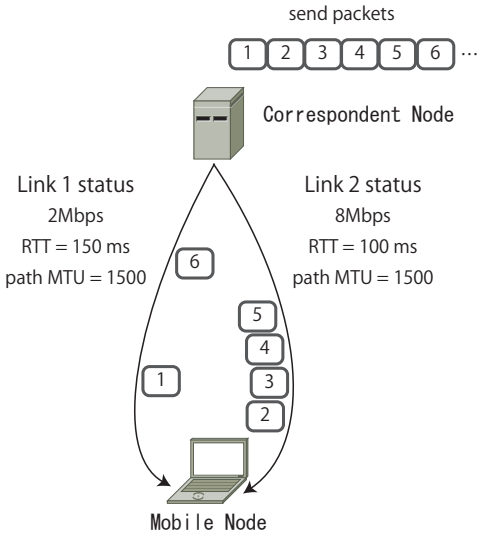

図 4: 異なる性質のパスにおけるマルチパス転送

- Round Trip Time (RTT)
  異なるパスにを用いた通信では，各パスの Round Trip Time (RTT) が異なると考えれる．図 4 では転送レートを帯域幅に応じて設定すると 1:4 の比率でデータが送信されるが，これらのパスは遅延が異なるため，delay-bandwidth product の比は 3:8 となる．そのため，各パスへのデータの送信レートは帯域幅だけでなく，遅延も考慮するべきである．図 4 の場合には，各パスへのデータ送信量は 3:8 の割合になるべきである．また，図 4 で Link 1 へパケット 1 が送信され，次に Link 2 へパケット 2 - 5 が送信されたとする．この際，それぞれのリンクは遅延が異なるため，Mobile Node で受信される順番はパケット 2 - 5 が先に受信され 25 ms 遅れてパケット 1 が到着する．パケットの到着順序の変更は輻輳制御メカニズムにより通信効率の低下を招くため，できる限り避ける必要がある．

以上で述べた点を考慮し，ARMS では取得可能なパスのパラメータとして Congestion Window (*cwnd*) [4] を使用する．*cwnd* は TCP および SCTP において送信側で管理される輻輳制御のパラメータで，SCTP においてこの値は宛先毎に独立している．*cwnd* は送信側が ACK の受信を待たずに送信可能なデータの量で，経路の帯域および遅延を反映している．理想値としては，delay-bandwidth product の値になる．

この *cwnd* をパス毎に取得しその値の比率に応じて転送比率を決定していく．例えば，Link 1 の *cwnd* が 20000 で，Link 2 の *cwnd* が 80000 であるとすると，送信比率は 1 : 4 となる．この送信比率でそれぞれの *cwnd* を埋めきるまで 図 4 のようにデータを送信する．両パスの *cwnd* を埋めきったところで再度各宛先に対する *cwnd* を取得して転送比率を再計算し，その比率に応じて送信を行う．

cwnd は SCTP が提供する API により取得可能である．取得方法は，*getsockopt()* 関数の第 3 引数にて *SCTP_GET_PEER_ADDR_INFO* を指定する．

4. 評価

本節では ARMS のプロトタイプ実装と，プロトタイプ実装による評価について述べる．プロトタイプ実装として，(1) まず通常の SCTP の *one-to-many socket style* による SCTP アソシエーションを確立するために，引数に *SOCK_SEQPACKET*，*IPPROTO_SCTP* を指定してソケットを生成する．(2) 次に，1 つのアドレスに対して *sctp_sendmsg()* 関数を実行しデータを送信する．その際，SCTP は内部で 1 つのアソシエーションを確立し，そのアソシエーションにはクライアント，サーバそれぞれの IP アドレスが含まれる．(3) その後 *sctp_getpaddrs()* 関数を用いて通信相手のアドレスリストを取得する．(4) 次に，*getsockopt()* 関数によって各宛先に対する *cwnd* を取得する．(5) 各宛先に対する *cwnd* がそれぞれ *cwnd1*, *cwnd2* であった場合，*sctp_sendmsg()* 関数を用いて宛先 1 に *cwnd1* Byte, 宛先 2 に *cwnd2* Byte データを送信する．その後は，(4), (5) を繰り返し行う．

今回のこのプロトタイプ実装における実験では 2 つのパスを同時に利用し，各パスの帯域幅と RTT を変化させて実験を行った．実験環境を図 5 に示す．Correspondent Node (CN)，Mobile Node (MN) ともに OS は FreeBSD 7.0 を使用した．また、CN は 1 つのネットワークに有線にて接続され，MN も 2 つのネットワークに有線で接続している．また CN と MN が接続している 3 つのネットワークは全て異なるセグメントである．Dummynet を用いて MN が接続する *Network 1* および *Network 2* の帯域と遅延を変化させる．また，CN が接続する *Network 3* は 100Mbps である．今回の実験では，帯域は 1Mbps または 2Mbps に設定し，RTT は 30ms または 60ms と設定した．尚，以降では，帯域が 1 Mbps で RTT が 30 ms のパスを *path(1Mbps, 30ms)*，帯域が 2Mbps で RTT が 60 ms のパスを *path(2Mbps, 60ms)* と表記する．これらにシングルパスまたは，マルチパスの組み合わせを含めた全 14 パターンにて，20MByte を転送するときに要した時間を計測した．全 14 パターンのネットワーク環境について，それぞれ 5 回づつ実験を行い，その結果を表 1 にまとめた．表 1 の縦軸 (*path1)* が図 5 の Network2 の設定を，表 1 の横軸 (*path2)* が の図 5 の Network 3 の有無または存在する場合の設定を示す．平均スループットは，データ転送量を総転送時間で割ることによって算出し，図 6 に示した．

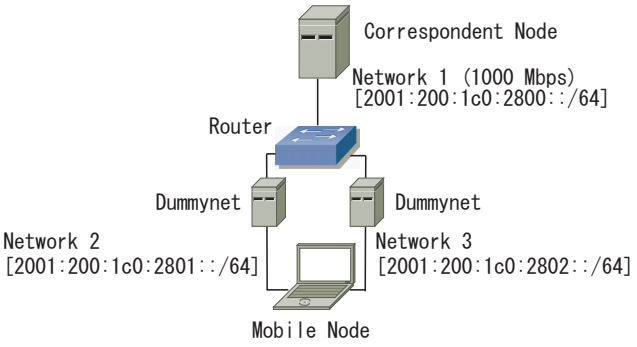

図 5: 評価環境

4.1 同じ性質のパスを用いた場合

path(1Mbps, 30ms)，*path(1Mbps, 60ms)*，*path(2Mbps, 30ms)*，*path(2Mbps, 60ms)* を単独で用いた際のスループットはそれぞれ 0.87Mbps, 0.84Mbps，1.69Mbps，1.60Mbps であった．一方で，ARMS によりそれぞれのパスを 2 本同時に利用した際のスループットは，*path(1Mbps, 30ms)* を二つ用いた場合が 1.65Mbps，*path(1Mbps, 60ms)* を二つ用いた場合が 1.63 Mbps，*path(2Mbps, 30ms)* を二つ用いた場合が 3.28 Mbps，*path(2Mbps, 60ms)* を二つ用いた場合が 3.20 Mbps だった．

単一のパスを用いた場合と ARMS により複数パスを用いた場合のスループットの変化の割合は，*path(1Mbps, 30ms)* の場合が 1.90 倍，*path(1Mbps, 60ms)* の場合が 1.94 倍，*path(2Mbps, 30ms)* の場合が 1.94 倍，*path(2Mbps, 60ms)* の場合が 2.00 倍である．このことから，二つのパスを効率的に利用し，理想的なスループットの改善が見られた．

4.2 帯域が異なるパスを用いた場合

path(1Mbps, 30ms) と *path(2Mbps, 30ms)* の組み合わせ，*path(1Mbps, 60ms)* と *path(2Mbps, 60ms)* の組み合わせを用いた場合のスループットは，それぞれ 1.80 Mbps, 1.80 Mbps であった．

単一パスを用いた場合のスループットは *path(2Mbps, 30ms)* の場合が 1.69 Mbps，*path(2Mbps, 60ms)* の場合が 1.60 Mbps であったことから，1 割ほどのスループット向上がみられた．しかし，これは理想値には届いていないため，アルゴリズム及び実装の改善が必要である．

4.3 遅延が異なるパスを用いた場合

path(1Mbps, 30ms) と *path(1Mbps, 60ms)* の組み合わ

*path*1	*path*2	none	1Mbps		2Mbps	
			rtt : 30ms	rtt : 60ms	rtt : 30ms	rtt : 60ms
1M bps	rtt : 30ms	175.78	92.51	—	—	—
	rtt : 60ms	180.74	92.61	93.82	—	—
2M bps	rtt : 30ms	90.48	84.85	84.78	46.45	—
	rtt : 60ms	95.26	84.81	84.79	47.15	47.61

表 1: 各パスの組み合わせにおける 20MByte の転送時間 (秒)

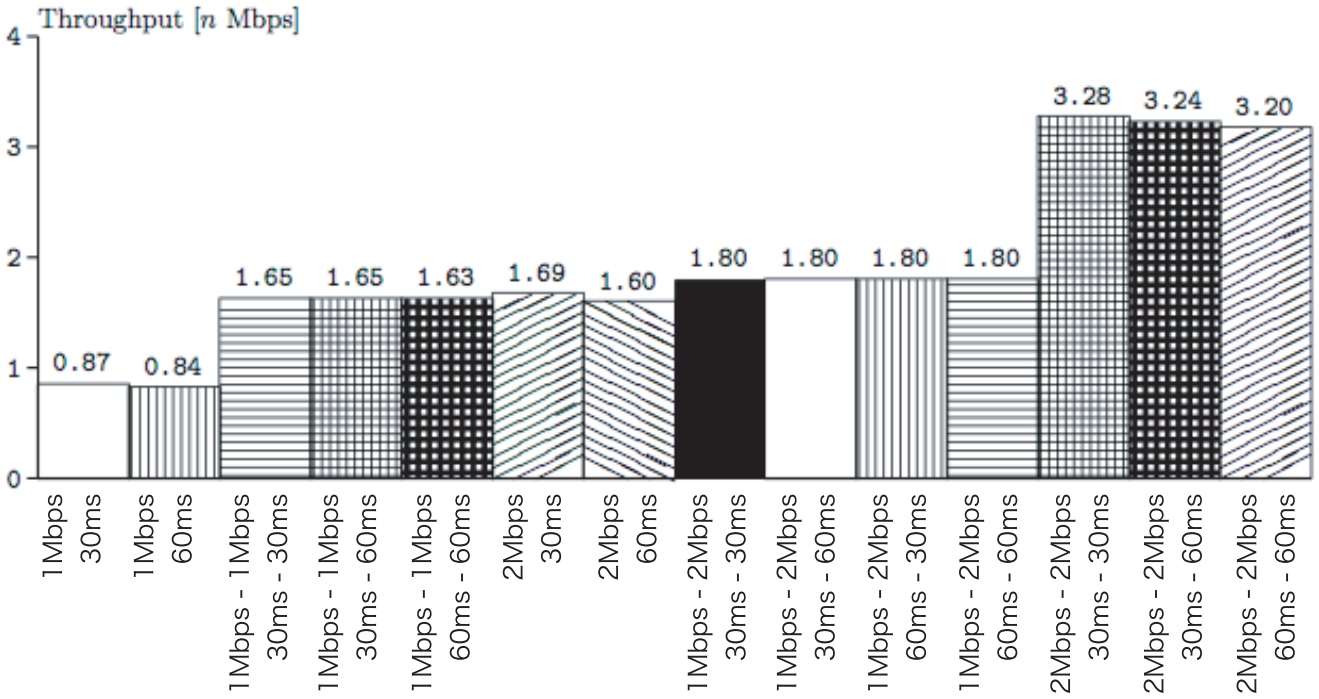

図 6: 平均スループット比較

せ，*path(2Mbps, 30ms)* と *path(2Mbps, 60ms)* の組み合わせを用いた場合のスループットは，それぞれ 1.65Mbps, 3.24Mbps であった．

path(1Mbps, 30ms) のシングルパスと，*path(1Mbps, 30ms)* と *path(1Mbps, 60ms)* の組み合わせのマルチパスではスループットが約 1.90 倍の差となり，*path(2Mbps, 30ms)* のシングルパスと，*path(2Mbps, 30ms)* と *path(2Mbps, 30ms)* の組み合わせのマルチパスではスループットが約 1.92 倍の差となった．この場合も帯域をうまく埋めきることができたため理想的なスループットの向上が見られた．

4.4 帯域及び遅延が異なるパスを用いた場合

path(1Mbps, 30ms) と *path(2Mbps, 60ms)* の組み合わせ，*path(1Mbps, 60ms)* と *path(2Mbps, 30ms)* の組み合わせを用いた場合のスループットは，それぞれ 1.80Mbps, 1.80Mbps であった．

帯域のみが異なるパスを用いたときと同様に，単一パスを用いた場合のスループットは *path(2Mbps, 30ms)* の場合が 1.69 Mbps，*path(2Mbps, 60ms)* の場合が 1.60 Mbps であったことから，1 割ほどのスループット向上がみられた．しかし理想的な改善には及んでいないため，アルゴリズム及び実装の改善が必要である．

5. まとめと今後の展望

本研究では複数のネットワークインタフェースを持つ端末のネットワーク資源を効率的に利用するため，アプリケーションレベルで複数パスを同時に用いた信頼性のある通信を実現する手法，ARMS を提案した．またプロトタイプを実装し，評価を行った．現在様々な複数パス

を同時に利用する手法が提案されているにも関わらずどれも実際には利用できない中で，ARMS はアプリケーションレベルで実現することによりプラットフォームに依存せずに利用できる．

また，ARMS はトランスポートプロトコルに SCTP を用いることにより，異なるパスを流れる複数のフロー間でのデータの同期をアプリケーションで行う必要がない．また，効率的な複数パス同時利用に必要となるパス毎の情報は SCTP の API によって取得できる．これらのことから，TCP を利用した場合に比べ，容易に複数パスを同時に利用した通信を実現できる．

評価として 2 つのパスを利用したデータ転送による実験を行った．その結果，帯域幅が同じ環境化ではほぼ理想的なスループットを実現できた．一方で帯域幅が異なる環境化では，多少のスループット向上は見られたものの理想的な改善には及ばなかった．そのため，今後の展望として実装を改善し，帯域幅が異なる場合の理想的なスループットの実現を計る．また，今回は 2 つのパスを利用したため，今後はより多くのパスを利用した場合の実験も行い，ARMS のパスの数に対するスケーラビリティを検証する．

参考文献

[1] Hung-Yun Hsieh and Raghupathy Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 83–94, 2002.

[2] Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Trans. Netw.*, 14(5):951–964, 2006.

[3] LuizMagalhaes and RobinKravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*, page 165, Washington, DC, USA, 2001. IEEE Computer Society.

[4] M. Allman, V. Paxson and W. Stevens. TCP Congestion Control. *RFC 2581*, Oct. 1999.

[5] Junwen Lai Ming Zhang and et al. A transport layer approach for improving end-to-end performance and robustness using redundant paths. *USENIX 2004 Annual Technical Conference*, pages 99–112, 2004.

[6] R. Stewart. Stream Control Transmission Protocol. *RFC 4960*, Sep. 2007.

[7] R. Stewart, Q. Xie and et al. Sockets API Extensions for Stream Control Transmission Protocol (SCTP). *Internet Draft*, Jul. 2008.

[8] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. *RFC 5061*, Sep. 2007.

[9] Kultida ROJVIBOONCHAI, Toru OSUGA, and Hitoshi AIDA. A study on rate-based multipath transmission control protocol (r-m/tcp) using packet scheduling algorithm(tcp protocol, ¡special section¿new technologies and their applications of the internet iii). *IEICE transactions on information and systems*, 89(1):124–131, 20060101.

[10] Shunsuke Saito, Yasuyuki Tanaka, Mitsunobu Kunishi, Yoshifumi Nishida and Fumio Teraoka. AMS: An Adaptive TCP Bandwidth Aggregation Mechanism for Multi-homed Mobile Hosts. *IEICE Transactions on Information and Systems*, 89:2838–2847, 2006.