ネットワーク機能を追加できる透過型代理通信システムの実装とその評価

石田正人[†1] 中野宣昭[†2] 桝田秀夫[†3]

近年，インターネットやパソコンが普及するにつれて，大学や家庭内においてもネットワークが構築されるようになった．それにより，家電製品などの今までネットワークに対応していなかった機器がネットワーク機能を持つようになり始めた．

機器がネットワークに対応することで，それを利用した機器の一元管理が可能となることが期待される．例えば，ネットワーク機器を監視や制御するためのプロトコルである SNMP（Simple Network Management Protocol）を用いた機器管理である．これを利用することで比較的簡単にネットワーク経由で機器を管理することができる．しかし，全てのネットワーク対応機器が SNMP に対応しているわけではない．その場合，対応していない機器については，別の管理方法を考える必要があるが，可能であれば，他の対応している機器と同様に管理できることが望ましい．

そこで本論文では，機器に実装されていないネットワーク処理の機能を別のシステムが代わりに補うことで，その機器が対応しているかのように振る舞うことを可能とするシステムを提案する．このシステムは，対象となる機器と他の通信機器との間に設置する透過型代理通信システムとして実装している．その際，他の通信機器は，システムを設置することによって，ネットワーク設定などの変更といった影響を与えることなく，システムを設置する前の状況と同様に使用することが可能となる．また，対象の機器に実装されていないネットワーク処理が必要となった場合は，このシステムが代理で処理を実行したり，応答したりすることで透過的に対応する．

このシステムを Linux を用いて実装し，スループットを計測した結果，Pentium III 1GHz クロック程度の CPU を持つ PC 上で kernel bridge と同程度の約 5% 程度の速度低下で実現できた．

Implementation and Evaluation of Transeparent Proxy System of Addable Network Functions

Masato Ishida,[†1] Nobuaki Nakano[†2] and Hideo Masuda [†3]

Many devices are connected via network not only in University, Company but also Home in recent years. Thereby, home electronics began to have a network function. We expect that all network-connected devices are managed by the unified management system via standard function such as SNMP. However some devices don't have such a network management function, but it is desirable that all devices become manageable. In this paper, we describe the transparent proxy system of addable network functions. This system enables that wanted network functions can be added into any network-connected device without additional configurations. We also show the performance evaluation, and we establish the speed fall less than 5 % against kernel bridge on Linux PC which has Pentium III 1GHz clock CPU.

1. はじめに

近年，インターネットやパソコンが普及するにつれて，大学や家庭内においてもネットワークが構築されるようになった．それにより，家電製品などの今までネットワークに対応していなかった機器がネットワーク機能を持つようになり始めた．それは，パソコンの周辺機器であるプリンターやハードディスク，また，テレビやハードディスクレコーダーなどの家電製品がある．他にはオンラインゲームの普及により家庭ゲーム機がネットワークに対応するようになった[1)]．

機器がネットワークに対応することで，それを利用した機器の一元管理が可能となることが期待される．例えば，ネットワーク機器を監視や制御するためのプロトコルである SNMP（Simple Network Management Protocol）を用いた機器管理である．これを利用することで比較的簡単にネットワーク経由で機器を

†1 特定非営利活動法人 レーザー技術推進センター
Promotion Center for Laser Technology
†2 京都大学 大学院情報学研究科
Graduate School of Informatics, Kyoto University
†3 京都工芸繊維大学 情報科学センター
Center for Information Science, Kyoto Institute of Technology

管理することができる．しかし，全てのネットワーク対応機器が SNMP に対応しているわけではない．その場合，対応していない機器については，別の管理方法を考える必要があるが，可能であれば，他の対応している機器と同様に管理できることが望ましい．

他にも，ネットワークのセキュリティ向上のため，IEEE 802.1x や Web アクセスを使用して，利用者認証を行って接続の可否やアクセス範囲の限定を行う仕組みがある．これは，あらかじめ決められた機器以外がネットワークに参加することを認証によって制限する機能である．そのため，装置がこの認証方式に対応していない場合は，ネットワークに参加することができない．

例えば広島大学では，大規模なキャンパスネットワークを構築し，そのネットワーク内の通信機器は Web 認証を行うことでネットワークに参加する仕組みが構築された[2)]．Web 認証を利用することでブラウザがインストールされたパソコンであれば，OS（Operating System）の種類に関係なく認証を行うことができる．

しかし，ルータやネットワークプリンタなどのブラウザがインストールされていない機器の場合は，Web 認証が行えない．これを補うために Web 認証とは別に MAC（Media Access Control）アドレス認証を設置している．これにより，認証を行うサーバが 2 種類必要となること．また，MAC アドレス認証ならではの問題に対応する必要性が発生した[3)]．

そこで本論文では，機器に実装されていないネットワーク処理の機能を別のシステムが代わりに補うことで，その機器が対応しているかのように振る舞うことを可能とするシステムを提案する．これにより，機器が対応していないために起こる問題を解消することを目的とする．提案するシステムを透過型代理システムとする．

このシステムを設置することで機器に実装されていないネットワーク処理を対象の機器や他の通信機器に変更を加えることなく，容易に追加することができる．

以降，2 章では透過型代理システムへの要求と仕様について述べる．3 章では，システムの仕様書について述べ，4 章では，システムの実装について述べる．5 章では，本研究に対する評価を述べ，6 章で本論文のまとめを述べる．

2. 要　　求

本章では，透過型代理システムに対する要求について述べる．

R1　ネットワーク透過性を有すること
まず，本システムは，できるだけその存在を主張せず，利用者機器のネットワークパラメータのみで動作することが望ましい．例えば，通信に利用する IP アドレスや MAC アドレスは，利用者機器のもののみがネットワーク中に存在するようにすれば，ネットワーク管理者が利用者機器の素性を知る際に OUI からベンダーの類推が可能となることや，ネットワーク管理者と利用者が認識する IP アドレスが同一となることで意志の疎通がはかりやすい．

R2　設定が自動化されていること
また，本システムへのネットワークパラメータの設定は自動的になされることも望まれる．利用者機器へのネットワーク設定を利用者が変更する際に，本システムも同時に設定を変更する必要があるとなると，利便性が低下し，トラブルも増加することが予想される．

R3　低コストで実現できること
さらに，本システムはできるだけ低コストで実現できることも望まれる．SNMP 管理ができない機器や IEEE802.1x に非対応の機器は，概ね安価な機器であることが多いため，本システムが非常に高価となるようでは，全体の運用コストの削減への寄与が減少してしまう．文字通り，透過的に動作するように，できるだけスループットを下げない工夫が求められる．

3. 検　　討

本システムがネットワーク処理を代理で応答するためには，利用者機器に送信されるパケットをすべて受信する必要がある．そのために対象の機器と他の通信機器との間に設置することでこれを実現する．また，他の通信機器に影響を与えることなくシステムを設置するために透過的なブリッジ動作を行うものとする．

対象となる利用者機器として応答するために必要となる情報として，IP アドレスや MAC アドレスがある．この情報を手動で設定するのではなく，自動で情報を収集，設定する仕組みを実現するためにやりとりされるパケットのヘッダ情報を常に監視する．

追加するネットワーク処理としては，ここでは UDP（User Datagram Protocol）と TCP（Transmission Control Protocol）が用いられた処理を対象とする．

対象となる利用者機器側の通信が切断された場合には，システムは利用者機器の情報を初期化することで状態の変化に追従する機能も実現する．

4. 実　　装

本章では，透過型代理システム（以下，本システムと呼ぶ）の実装について述べる．

本システムの構成を図 1 に示す．

本システムを実装する対象の OS として比較的低スペック環境における動作が可能であり豊富なネットワーク処理ができる Linux を選択した．

実装するプログラム言語としては，高速な動作と低

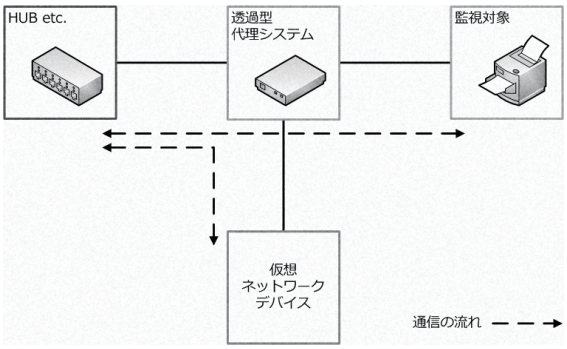

図 1 実装するシステムの構成

レベルなネットワークへのアクセスを容易に行うために C 言語を選択した．

実装対象とする装置については，ブリッジ動作を行うため，イーサネットデバイスが 2 つあるものを使用した．

4.1 透過的なブリッジ動作

まず，ブリッジ動作を行うためには，UDP や TCP，ARP（Address Resolution Protocol）などの全ての通信プロトコルに対応する必要がある．後述する利用者機器に関するネットワーク情報の自動取得や代理応答を行うためには，パケット内のイーサネットヘッダや IP ヘッダといった部分を扱えるようにしておかなければならない．

それらの条件を実現するために通信に使用するソケットは，LPF（Linux Packet Filter）で使用されている RAW ソケットを用いた．RAW ソケットを用いることで TCP ソケットなどでは扱えないデータリンク層で扱うようなパケットをユーザ空間で扱うことを可能とする．これにより，プロトコルに関係なくプログラム内でパケットの操作や解析を行うことができる．

また，通常のネットワークデバイスは，宛先が自分宛になっているパケットのみを受信するように設定されている．自分宛以外のパケットについては届いても読み込まずに破棄するようになっているため，自分宛以外のパケットも扱う設定としてプロミスキャスモードを利用する．

動作のシーケンスをまとめると下記の通りとなる．

(1) 使用する通信ソケットを RAW ソケットで作成する．
(2) 使用するネットワークデバイスをプロミスキャスモードで動作するように設定する．
(3) ネットワークデバイスを作成したソケットにバインドする．
(4) 各ソケットの受信状態を監視する．
(5) 監視していたソケットがパケットを受信すると，
  (a) 他の通信機器側からのパケットを受信した場合には，受信したパケットを利用者機器側のソケットに送信する．
  (b) 利用者機器側からのパケットを受信した場合には，受信したパケットを他の通信機器側のソケットに送信する．

以上の動作を行うことで本システムが受信したパケットは，特に変更を加えることなく，送信先に送信される．つまりは，透過的なブリッジ動作を実行していることとなる．

4.2 利用者機器に関するネットワーク情報の自動取得

代理応答を行う際に必要となる利用者機器のネットワーク情報は，ブリッジ動作により得られるパケットから取得する．

利用者機器から送信されるパケットを全て読み取ると，パケット内のイーサネットヘッダから利用者機器の MAC アドレス，IP ヘッダから利用者機器の IP アドレスを取得することができる．

取得した利用者機器の情報に関しては，利用者機器が複数接続されていた場合でも対応できるようにするため，双方向リストに格納するようにした．このリストを対象機器管理リストとする．

動作シーケンスは下記の通りである．

(1) 利用者機器側からのパケットを受信するソケットを監視対象とする．
(2) 受信したパケットから IP アドレスと MAC アドレスを取得する．
(3) 取得したアドレス情報を対象機器管理リストから検索し，重複した情報があるかどうかを確認する．
(4) 重複した情報が無ければ，対象機器管理リストに対して新規に取得したアドレス情報を追加する．

このようにして，利用者機器の設定情報を，通信を見張ることにより自動でネットワーク情報を取得する動作を実装した．

4.3 代理応答を行うための設定

代理応答を行うために必要となる利用者機器に関するネットワーク情報の取得に関して，前節で述べた．本節では，その情報を用いて代理応答するために必要な設定について述べる．

代理で応答するということは，本システムから送信するパケットの送信元情報が利用者機器のネットワーク情報に設定されている必要がある．

通常，本システムから特別な設定をせずにパケットを送信するとパケットには本システムのネットワーク情報が送信元に設定される．それを利用者機器を送信元に変更するには，パケット内のイーサネットヘッダや IP ヘッダの情報を書き換え，FCS（Frame Check Sequence）といったチェックサム符号を計算し，適切な値に書き換えることになる．

パケット内容の書き換えの動作を実装することは可能であるが，チェックサム符号の再計算などの処理を

ユーザランドプログラムで処理するとオーバヘッドが大きいことが予想される．この問題を解決するためにTUN/TAP デバイスドライバを利用した．

TUN/TAP デバイスドライバを使うことで，仮想ネットワークデバイスを作成することができる．OSは仮想ネットワークデバイスが受信したパケットを物理デバイスが受信したときと同様の処理をカーネルレベルで実行する．そのため，仮想ネットワークデバイスに対して，取得した利用者機器の IP アドレスとMAC アドレスを設定しておけば，当該デバイスが受信，処理を実行後，返送する際に送信されるパケットの送信元は設定したネットワーク情報が設定される．これにより，ユーザランドプログラムで独自にパケットの内容を書き換える手間がなくなる．

仮想ネットワークデバイスに対する設定の流れは下記の通りである．

(1) 対象機器管理リストに対して新規にアドレス情報が追加される．
(2) 仮想ネットワークデバイスを作成する．
(3) 作成したデバイスに対して，新規に追加されたアドレス情報を設定する．

4.4 代理応答の動作例

本システムが利用者機器の代理システムとして，ネットワーク処理を実行する際の動作モードは 2 種類考えられる．1 つは本システムがサーバとしての処理を実行することであり，もう 1 つはクライアントとしての処理を実行することである．

4.4.1 代理応答: サーバモード

まず，サーバモードの処理の実例として，本システムに SSH サーバと SNMP エージェントを実装した．SSH サーバは，OpenSSH_5.1p1[4] の sshd，SNMP エージェントは，Net-SNMP[5] version.5.4.1 の snmpd をそれぞれ選択した．

SSH サーバを実装することで本システムが稼働している時に外部から本システムへのアクセスが可能となる．また，SNMP エージェントを実装することで SNMP を利用した機器管理に対応することができる．

ここで想定している使用方法は，通過する通信トラヒックだけではなく，代理システムがハードウェア I/O を有して監視対象に対して物理的なセンサーを用いて情報を SNMP MIB に載せたり，/proc デバイス経由で参照できるようにすることが考えられる．

通信のプロトコルとして，SSH サーバは TCP，SNMP エージェントは UDP で動作するため，複数のプロトコルでの動作検証を行うことができたと言える．

サーバの応答動作に必要な設定として，あらかじめサーバが受信に使用するポートとプロトコルをシステムに設定しておく．設定した情報は対象機器管理リストと同様に複数の情報を設定できるようになっており，これをインターセプト情報リストとする．このリストに設定したプロトコル及びポート宛のパケットを本システムが受信した際にサーバが応答する動作を実行するようになっている．

代理応答の動作シーケンスは以下の通りである．

(1) インターセプト情報リストに使用するポートとプロトコル番号を設定する．
(2) 他の通信機器側からのパケットを受信するソケットと仮想ネットワークデバイスのソケットを監視対象とする．
(3) 他の通信機器側からパケットを受信した際に宛先 IP アドレスと MAC アドレスを取得する．
(4) 取得したアドレス情報が対象機器管理リストから検索し，適合する情報が存在するかどうかを確認する．
(5) 取得した情報がリストに存在していた場合は，パケットからさらに宛先ポート番号とプロトコル番号を取得する．存在しない場合は，ここで処理を終える．
(6) 取得した情報をインターセプト情報リストから検索し，適合する情報が存在するかどうかを確認する．
(7) 取得した情報がリストに存在した場合には，取得した宛先 IP アドレスに該当する仮想ネットワークデバイスにパケットを送信する．存在しない場合は，ここで処理を終える．
(8) サーバーが処理を実行し，クライアントに対して返送するパケットが発生する．
(9) サーバーから送信されるパケットは仮想ネットワークデバイスのソケットから読み取ることで取得でき，このパケットを他の通信機器側へのソケットに送信する．

4.4.2 代理応答: クライアントモード

次にクライアントの処理として，本システムに SSH クライアントを実装した．SSH クライアントは，OpenSSH_5.1p1 の ssh を選択した．

SSH クライアントによる通信を可能とすることで，SSH による接続を元に接続認証を許可するような認証付きネットワークシステムに適用できる．

サーバモードの場合は，本システム側への通信に使われる送信先ポート番号はサーバに対して固定であるが，クライアントモードでの動作の場合，通信に使用するローカルポートが動的に設定されることになる．そのため，送信先からの返信されたパケットを本システム宛であるかどうかを判断するために必要となるポート番号を通信開始時に取得する機能が必要となる．

これには，代理応答で使用される仮想ネットワークデバイスも監視対象とすることで，送信されたパケットの内容より，送信元のポート番号を取得するようにした．

代理応答の動作シーケンスは以下の通りである．

(1) 他の通信機器側からのパケットを受信するソケットと仮想ネットワークデバイスのソケットを監

視対象とする．
(2) システムがクライアント動作を実行するシェルスクリプトなどを呼び出して，他の通信機器に通信を試みる．
(3) クライアント動作時には，送信元を対象の機器の設定をした仮想ネットワークデバイスに設定しておく．
(4) 他の通信機器に通信を試みる際に発生するパケットを仮想ネットワークデバイスのソケットから読み取ることで取得する．
(5) 取得したパケットから送信元のポート番号とプロトコル番号を取得する．
(6) 取得したポート番号などをインターセプト情報リストから検索し，重複する情報が存在するかどうかを確認する．
(7) リストに情報が存在しない場合には，取得した情報をリストに追加する．
(8) 取得したパケットを他の通信機器側へのソケットに送信する．
(9) サーバーの応答動作の 3〜9 と同様な動作を実行することで応答する．

4.5 利用者機器側の通信が切断された際の動作

利用者機器側の機器を停止した場合には，透過性の観点から，それを検知して動作をやめる必要がある．

そのため，必要の無くなった情報を削除する必要性が発生する．利用者機器の情報を適切なタイミングで削除するためには，通信が切断されたことを検出する必要がある．

検出する方法として，イーサネットケーブルの接続状態を取得して反映させる．

利用者機器側の通信が切断された際の動作シーケンスは下記の通りである．

(1) 利用者機器側の通信が切断されたことを検知する．
(2) 対象機器管理リストの情報を全て削除する．
(3) 利用者機器に対する設定をした仮想ネットワークデバイスを全て破棄する．
(4) インターセプト情報リストから自動で取得した情報については，全て削除する．

5. 評　　価

本章では，実装した本システムについての評価について述べる．

5.1 実 行 環 境

実行環境の構成を図 2 に示す．

使用した機器の情報については，表 1 に示す．項目にある NIC（Network Interface Card）には，ネットワークインターフェースが対応する通信速度を示している．表中に情報のない HUB については，1Gbps, 100Mbps の通信速度に対応した機器を使用している．

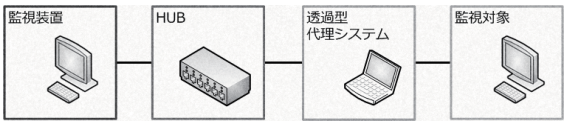

図 2 評価を行った機器の構成

本システムの性能が，どの程度低性能の PC にまで耐えられるかを評価するために，3 種類の PC を用意している．

5.2 評 価 方 法

システムの評価には，システムを用いたブリッジ動作におけるスループットの測定を測定した．また，システムがブリッジ動作を実行しているときの実行している機器の負荷を測定した．

5.2.1 スループットの測定

スループットの測定には，iperf[6] を使用した．

iperf は，ネットワークのパフォーマンスを測定するプログラムである．使用する際には，送信するデータのサイズや送信数などを設定することができる．通信に使用するプロトコルを TCP，UDP のどちらかを選択できる．

スループット測定は下記のコマンドで行った．サイズの部分には，1024, 2048, 4096, 8192 の 4 通りで測定を行った．

監視装置側：`iperf -s -l` サイズ
監視対象側：`iperf -c IPaddress -l` サイズ `-y c`

また，本システムの性能と比較のためにシステムを実行している機器に対して，本システムの代わりに bridge-utils を使ったカーネルによるブリッジのスループット測定を測定した．あと，本システムを実行している機器を省いたブリッジ動作の必要がない環境（直接接続）での測定も実施した．

5.2.2 ブリッジ動作実行時の負荷の測定

負荷測定には，vmstat を使用した．

vmstat を使用することでメモリや CPU の負荷率などの情報を表示することができる．

負荷測定は下記の通りに行った．

- vmstat を 1 秒ごとに情報を保存する設定で実行する．
- 10 秒程度経過した後に，iperf による計測を開始する．
- iperf が終了した後 10 秒程度待ってから，vmstat の記録を停止する．

5.3 結　　果

スループットの測定結果をグラフにしたものを図 3 に示す．上から順番に，表 1 のスペック A,B,C のものである．

このグラフより，スペック B および C の機器であれば，カーネルによるブリッジに比べ本システムによるブリッジは，ほとんどスループットが低下していないこと分かる．これは，システムによるブリッジ動作

表 1 評価を行った機器の情報

種類	OS	CPU	メモリ	NIC
監視装置	Arch Linux	Atom N270@1.6GHz	1GB	1000baseT
中継システム				
スペック A	Vine Linux 4.2 (i586)	Pentium MMX@200MHz	256MB	100baseTX
スペック B	Fedora 13 (i686)	Pentium III@935MHz	512MB	100baseTX
スペック C	Fedora 13 (i686)	AMD64 3500+@2.2GHz	4GB	1000baseT
監視対象	Vine Linux 5.1	AMD Sempron@1.05GHz	2GB	1000baseT

によって，ネットワークに与えるスループット低下の影響が少ないことを意味する．ただし，スペック A の機器では，半分程度のスループットの低下が見られた．表を見ても分かる通り，スペック A の機器は PentiumMMX 200MHz 程度の CPU 性能であり，スペック B の PentiumIII 1GHz 程度の CPU 性能ですでに遜色ないスループットが得られていることから，最近の機器であれば大きな問題は出ないものと予想される．

また，負荷測定を行った結果をグラフにしたものを図 4 に示す．

このグラフより，本システムによるブリッジは，カーネルのブリッジに比べかなり負荷が高いことが分かる．システム時間の CPU 使用率に関しては，最も CPU 性能の高いスペック C であっても，2 割程度を占有している．システム時間とユーザ時間を比較すると，多くがシステム時間で占有されていることも見て取れる．従って，CPU 専有の原因としては，RAW ソケットを使用することによるオーバヘッドが考えられる．逆に，ユーザ時間の CPU 使用率から，システムによるパケットの解析などによる負荷は，それほど大きくないことが分かった．

6. ま と め

本論文では，通信機器にネットワーク機能を追加するための透過型代理通信システムの実装を行った．

本システムを設置することで他の通信機器に影響を与えないようにするためにブリッジの動作にパケット分別機能を付与し代理でネットワーク機能を実行するための利用者機器のアドレス情報を自動で取得するようにした．これらにより，本システムが対象の機器の代理でネットワーク機能を透過的に実行する仕組みができたと考えられる．また実装したシステムは，概ね満足できる転送性能を維持したままネットワーク機能を代理できているという評価結果も得られた．

なお，今回は IPv4 で実装しているが，Ethernet フレームをそのまま読み書きし，IP ヘッダおよび TCP/UDP ヘッダは，やり取りされているフレームから抜き出している構成であることから，IPv6 であっても同様に実装可能であると考えられる．さらに，無線 LAN 接続についても，Ethernet フレームの生成ができれば，イーサネットコンバータのような形での実装が考えられる．

今後の課題としては，RAW ソケットを使用することによる CPU 使用率が高いことについて，より低コストで本システムを実装するためにはこれ抑えることが必要となる．そのためには，システムを kernel level のコードとして実装するといったことが考えられる．

参 考 文 献

1) 三菱 UFJ リサーチ&コンサルティング (株),「情報家電機器及びサービスの普及促進に関する調査」, 平成 20 年度我が国 IT 利活用に関する調査研究事業 (2008).
2) 田島, 近堂, 岸場, 大東, 岩田, 西村, 相原:「大規模キャンパスネットワークにおける MAC アドレス認証の管理手法」, 情処学会 IOT シンポジウム 2008, pp.121–130 (2008).
3) 近堂, 田島, 岸場, 大東, 岩田, 西村, 相原:「利用者認証機能を備えた大規模キャンパスネットワークの性能評価」, 情処学会 IOT 研究会, 2009-IOT-4, pp.265–270 (2009).
4) OpenSSH (オンライン), 入手先 `http://www.openssh.org/ja/` (参照 2009-12-14).
5) Net-SNMP (オンライン), 入手先 `http://www.net-snmp.com/` (参照 2010-01-11).
6) iperf (オンライン), 入手先 `http://iperf.sourceforge.net/` (参照 2011-01-31).

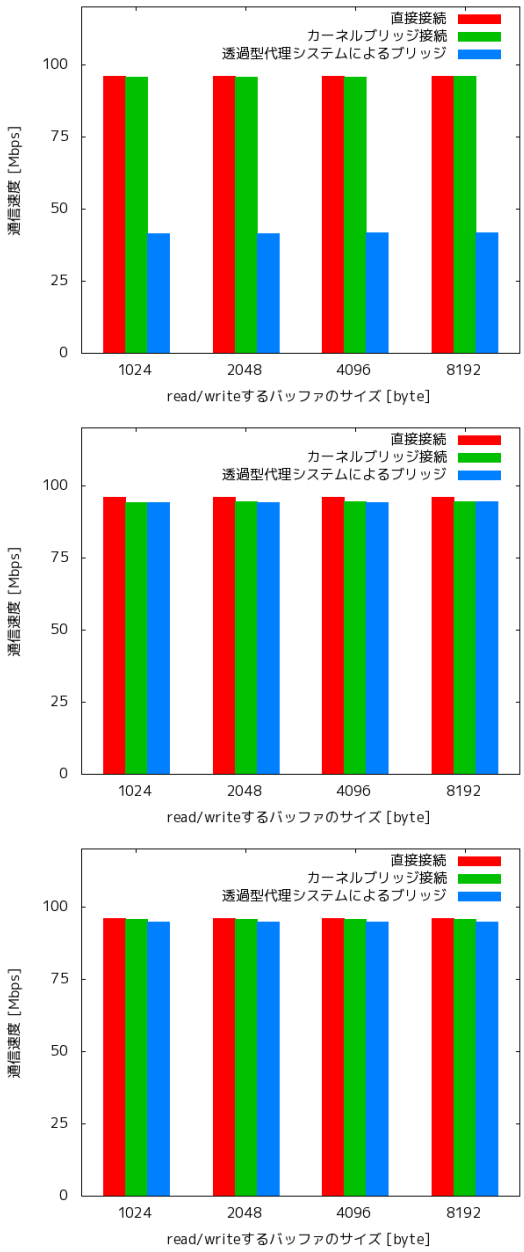

図 3 iperf によるスループット測定結果（TCP）

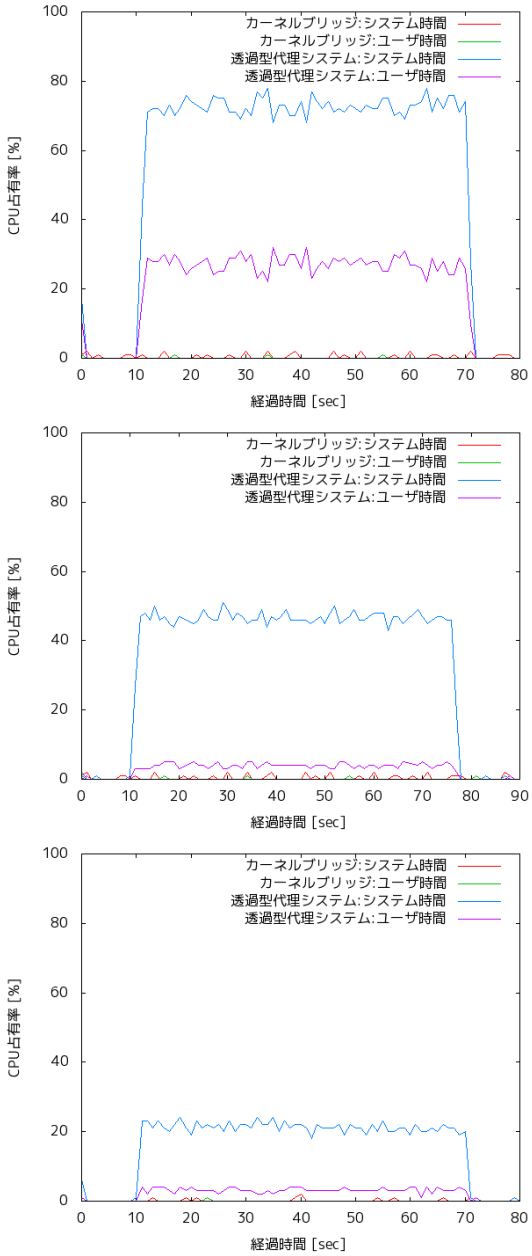

図 4 ブリッジ動作における CPU 使用率の測定結果