(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2009-27303
(P2009-27303A)
(43)公開日　平成21年2月5日(2009.2.5)

(51)Int.Cl.	F I			テーマコード（参考）
H04L 12/56 (2006.01)	H04L	12/56	200Z	5K030
	H04L	12/56	200D	

審査請求 未請求　請求項の数 4　OL　（全 30 頁）

(21)出願番号　特願2007-186566 (P2007-186566)
(22)出願日　平成19年7月18日 (2007.7.18)

特許法第３０条第１項適用申請有り　平成１９年２月９日　国立大学法人　電気通信大学主催の「平成１８年度　電気通信大学　卒業論文発表会」に文書をもって発表

特許法第３０条第１項適用申請有り　２００７年４月１２日　社団法人　電子情報通信学会発行の「電子情報通信学会技術研究報告　信学技報　Ｖｏｌ．１０７　Ｎｏ．６」に発表

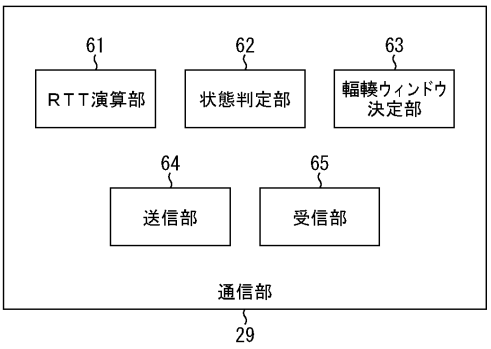
(71)出願人　504133110
国立大学法人　電気通信大学
東京都調布市調布ヶ丘１丁目５番地１
(74)代理人　100082131
弁理士　稲本　義雄
(74)代理人　100121131
弁理士　西川　孝
(72)発明者　塩津　晃明
東京都調布市調布ヶ丘１丁目５番地１　国立大学法人　電気通信大学内
(72)発明者　阿部　公輝
東京都調布市調布ヶ丘１丁目５番地１　国立大学法人　電気通信大学内

Ｆターム(参考)　5K030 GA03　GA13　HA08　JA07　LC03　LC09　LC11

(54)【発明の名称】通信装置および通信方法

(57)【要約】
【課題】他のコネクションとの利用帯域の公平性を保持しつつ、利用可能帯域を十分に利用する。
【解決手段】輻輳ウィンドウ決定部６３は、直前の輻輳ウィンドウcwndとIRTTの履歴を用いて、新たな輻輳ウィンドウcwnd´を予測する。送信部６４は、予測された輻輳ウィンドウcwnd´を用いて、インターネットの帯域を割り当て、その帯域でデータを送信する。
本発明は、例えば、インターネットを介した通信を行う通信装置に適用することができる。
【選択図】図４

図4



【特許請求の範囲】
【請求項１】
ネットワークを介した通信を行う通信装置において、
直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測する予測手段と、
予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信する送信手段と
を備える通信装置。
【請求項２】
前記予測手段は、前記直前の輻輳ウィンドウと、前記RTTの逆数の履歴を用いて、新たな輻輳ウィンドウを予測する
請求項１に記載の通信装置。
【請求項３】
前記予測手段は、前記直前の輻輳ウィンドウと、前記RTTの逆数の履歴を用いて、新たな輻輳ウィンドウを線形予測する
請求項２に記載の通信装置。
【請求項４】
ネットワークを介した通信を行う通信装置の通信方法において、
直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測し、
予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信する
ステップを含む通信方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、通信装置および通信方法に関し、特に、他のコネクションとの利用帯域の公平性を保持しつつ、利用可能帯域を十分に利用することができるようにした通信装置および通信方法に関する。
【背景技術】
【０００２】
近年、インターネットが普及してきている。また、それに伴いネットワークは複雑化し、輻輳の発生も増大している。輻輳は転送性能の低下を招くため、輻輳を回避する技術は重要である。
【０００３】
例えば、現在、TCP（Transmission Control Protocol）が、事実上、標準のインターネットプロトコルとなっている。しかしながら、TCPでは、中間ノードを透明化して通信を行うため、中間ノードは、流れてくるデータの全体量を知ることができない。これにより、
受信データをバッファから溢れさせてしまう現象、即ち輻輳が発生する。
【０００４】
輻輳によって喪失したデータの再送が行われると、そのデータによってまた輻輳が発生する輻輳崩壊という悪循環に陥るため、輻輳は悪化していく傾向がある。従って、輻輳を回避するために、輻輳を制御することが必要となる。
【０００５】
そこで、現在、TCPでは、パケットの喪失を検出すると、ウィンドウサイズを小さくすることで輻輳回避を行っている。具体的には、現在普及している、Renoというバージョンの TCPでは、図１に示すように、スロースタート状態１１と輻輳回避状態１２の２つの状態で、ウィンドウサイズを制御している。
【０００６】
図１のスロースタート状態１１では、データの確認応答であるACKが受信されるたびに

、輻輳ウィンドウcwndが１セグメント分増加する。即ち、スロースタート状態１１では、輻輳ウィンドウcwndが指数的に増加する。一方、輻輳回避状態１２では、ACKが受信されるたびに、輻輳ウィンドウcwndが、1/cwndセグメント分増加する。即ち、輻輳回避状態１２では、輻輳ウィンドウcwndが線形的に増加する。

【０００７】

また、輻輳ウィンドウcwndが閾値ssthreshより大きい場合、状態が、スロースタート状態１１から、輻輳回避状態１２に遷移する。なお、閾値ssthreshの初期値は、ウィンドウサイズの最大値であり、輻輳が検出された場合、そのときのウィンドウサイズの半分に設定される。

【０００８】

さらに、タイムアウトにより輻輳が検出された場合、そのときの状態が輻輳回避状態であればスロースタート状態に遷移し、そのときの状態がスロースタート状態であれば、その状態を維持する。また、重複ACKにより輻輳が検出され、かつ、再送が成功した場合、そのときの状態がスロースタート状態であれば、輻輳回避状態に遷移し、そのときの状態が輻輳回避状態であれば、その状態を維持する。

【０００９】

以下に、具体的な動作について説明する。

【００１０】

まず最初に、状態はスロースタート状態１１に設定され、輻輳ウィンドウcwndは最小値に設定される。そして、ACKが受信されるたびに、輻輳ウィンドウcwndが１セグメント分増加する。従って、送信したデータ全てに対するACKが受信されると、次に同時に送り出されるデータ量は、前回の２倍になる。その結果、短時間のうちに、利用可能帯域以上のデータ量のデータが送信されるようになり、輻輳が発生する。

【００１１】

そして、タイムアウトにより輻輳が検出されると、閾値ssthreshが、そのときのウィンドウサイズの半分に設定され、輻輳ウィンドウcwndが最小値に戻される。輻輳ウィンドウcwndは再び増加し、閾値ssthreshを越えたとき、状態が、スロースタート状態１１から輻輳回避状態１２に遷移する。

【００１２】

なお、このとき、輻輳が重複ACKにより検出された場合、パケットの再送に成功していれば、検出された輻輳が軽微なものであると判断され、輻輳ウィンドウcwndは最小値ではなく、そのときのウィンドウサイズの半分に設定される。従って、この場合、輻輳ウィンドウcwndが、直ぐに閾値ssthreshを超え、状態がスロースタート状態１１から輻輳回避状態１２に遷移する。その結果、輻輳ウィンドウcwndが最小値に戻されることによる、無駄な転送性能の低下を防止することができる。

【００１３】

輻輳回避状態１２では、輻輳ウィンドウcwndが、ACKが受信されるたびに、1/cwnd分増加する。従って、スロースタート状態１１の場合に比べて、輻輳ウィンドウcwndの増加速度が小さくなり、輻輳が発生するまでの時間が長くなる。しかしながら、輻輳ウィンドウcwndが減少することはないため、いずれ輻輳が発生する。そして、タイムアウトにより輻輳が検出されると、閾値ssthreshが、そのときのウィンドウサイズの半分に設定され、輻輳ウィンドウcwndが最小値に戻される。状態は輻輳回避状態１２からスロースタート状態１１に遷移する。

【００１４】

しかしながら、TCP Renoにおける輻輳回避は、輻輳発生後に行われるので、再送により余計なパケットを送信する必要があり、さらに、変化の激しいインターネットへの対応が困難である。また、輻輳が検出されるたびに、輻輳ウィンドウcwndが、そのときのウィンドウサイズの半分以下に低下するので、利用可能帯域を十分に活用することは難しい。

【００１５】

そこで、ネットワークの状態を予測し、輻輳を事前に回避する技術が研究されている。

この研究の手法は、大きく２つに分類され、１つ目の手法は、TCPではなく全く新しい高速で高信頼性のプロトコルを開発するものである。この手法では、新しいプロトコルが、広帯域高遅延を前提として設計されているため、この環境においては、従来のTCPより転送性能が高い。しかしながら、TCPとの互換性がないため、新しいプロトコルとTCPを仲介するための特殊なノードが必要となる。従って、新しいプロトコルを普及させるためには相当のコストがかかる。

【００１６】

また、２つ目の手法は、従来のTCPとの互換性を保持しつつ、輻輳回避アルゴリズムを改善するものである。この手法では、１つ目の手法のような劇的な転送性能の改善は難しいが、従来のTCPと互換性があるため、普及させるために特別な仲介ノードなどは必要ない。従って、輻輳回避アルゴリズムが改善されたTCPは、低コストで実装することができ、実用的である。

【００１７】

そこで、２つ目の手法により、Brakmoらは、VegasというバージョンのTCPを提案している。TCP Vegasでは、TCP RenoでRTO（Retransmission Time-Out）を算出するために使用されていたRTT（Round Trip Time）が正確に測定され、期待されるスループットと実際のスループットが計算される。そして、この２つのスループットを比較することで、ネットワークの輻輳が推測され、ウィンドウサイズが調整される。

【００１８】

具体的には、TCP Vegasにおけるスロースタート状態と輻輳回避状態での輻輳ウィンドウcwndの制御方法は、以下の式（１）で表される。

【００１９】

【数１】

$$cwnd = \begin{cases} [\text{スロースタート状態}] \\ \quad cwnd + \frac{1}{2} \\ [\text{輻輳回避状態}] \\ \quad \begin{cases} cwnd+1 & (Diff < \alpha / BaseRTT) \\ cwnd & (\alpha / BaseRTT < Diff < \beta / BaseRTT) \\ cwnd-1 & (\beta / BaseRTT < Diff) \end{cases} \end{cases} \quad \cdots (1)$$

【００２０】

なお、式（１）において、αとβは定数であり、定数αは定数βより小さい。また、BaseRTTは、これまでに測定された最小のRTTである。式（１）によれば、スロースタート状態において、輻輳ウィンドウcwndは指数的に増加する。

【００２１】

また、式（１）においてDiffは、期待されるスループットと実際のスループットの差であり、以下の式（２）で表される。

【００２２】

【数２】

$$Diff = \frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \quad \cdots (2)$$

【００２３】

以上のように、TCP Vegasでは、輻輳回避状態において、期待されるスループットと実

際のスループットの差により、ネットワークの状態が予測され、輻輳ウィンドウcwndが制御されるので、輻輳が軽微なネットワークにおいて、TCP Renoより転送性能を向上させることができる。

【００２４】

しかしながら、TCP Vegasでは、ネットワーク帯域が積極的に利用されるため、他のコネクションとの帯域利用の公平性が損なわれてしまう。また、変化の激しいネットワークでは、BaseRTTの測定が困難である。さらに、重度の輻輳では、輻輳回避アルゴリズムが機能しない可能性がある。また、TCP Renoでは、上述したように、輻輳ウィンドウcwndは増加するだけで減少しないので、TCP Vegasによる通信がTCP Renoによる通信と帯域を共有した場合、RTTが低下し、その結果、転送性能が著しく低下してしまう。

【００２５】

また、輻輳制御アルゴリズムが改善されたTCPとして、TCP Vegas とTCP Renoとで帯域を共有した際に転送性能が低下する問題を解決したもの、インライン計測により利用可能帯域の情報を取得し、その情報を利用したウィンドウサイズ制御アルゴリズムを有するものなどが提案されている。しかしながら、これらのTCPは、他のコネクションとの帯域利用の公平性の問題から、実用化には至っていない。

【００２６】

さらに、輻輳制御アルゴリズムが改善されたTCPとして、機械学習を取り入れることにより、ネットワークの状態から輻輳を予測し、ウィンドウサイズを制御する輻輳制御アルゴリズムを有するTCPも提案されている。このTCPでは、ネットワークの変化に素早く反応するなどの改善が確認されたが、帯域を積極的に利用するため、他のコネクションと帯域を共有した場合の利用帯域の公平性が犠牲になってしまう。

【００２７】

ところで、従来の輻輳制御方法としては、ATM（Asynchronous Transfer Mode）などの通信網において、実データとは別にRTT計測用のパケットを定期的に送信し、そのパケットのRTT時系列から、ニューラルネットを用いて次のRTTを予測し、予測したRTTにより転送レートを増減する自立分散型トラヒックフロー制御方法がある（例えば、特許文献１参照）。

【００２８】

しかしながら、この自立分散型トラヒックフロー制御方法では、RTT計測用のパケットを送信する必要があるので、パケットの転送量が増加してしまう。また、この自立分散型トラヒックフロー制御方法は、ATMなどの通信網における通信の制御を目的として考案されているため、他フローとの公平性は考慮されていない。

【００２９】

また、従来の輻輳制御方法としては、帯域の利用効率の改善、および、既存TCPとの帯域利用の公平性を目的として、ボトルネックルータのバッファ量を推定し、バッファを最大限に使うようにウィンドウサイズを制御する輻輳制御方法がある（例えば、特許文献２参照）。

【００３０】

しかしながら、この輻輳制御方法では、輻輳発生直前のDiff値(Diff_max)のみを用いる学習により、バッファ量を推定しているため、学習の精度は低い。そのため、輻輳を防止するためには、バッファ量を低めに推定せざるを得ず、利用可能帯域は十分に利用することが困難である。

【００３１】

さらに、従来の輻輳制御方法としては、マルチメディアデータのTCP通信のエンドツーエンドトラフィック管理を目的として、過去のパケット送信履歴と応答履歴で定まる回線の状態（キュー占有率）をキューでモデル化し、1時刻先からm時刻先までの応答時刻である予測応答時系列をマルチタイムスケール線形予測により予測することにより、その予測応答時系列とパケット送信履歴で定まる回線の状態を予測し、その予測された回線の状態に応じて、個々のパケットの送信時刻を決定する方法がある（例えば、特許文献３参照）

。
【００３２】
しかしながら、この方法では、予測応答時系列をマルチタイムスケール線形予測により予測するので、予測の精度は低い。そのため、輻輳を防止するためには、回線の状態のレベルを低めに予測せざるを得ず、利用可能帯域は十分に利用することが困難である。
【００３３】
【特許文献１】特開平１０－２３０６４号公報
【００３４】
【特許文献２】特開２００６－６７１０９号公報
【００３５】
【特許文献３】特開２００２－３６８８００号公報
【発明の開示】
【発明が解決しようとする課題】
【００３６】
以上のように、他のコネクションとの利用帯域の公平性を保持しつつ、輻輳を事前に回避することにより、利用可能帯域を十分に利用する輻輳制御アルゴリズムは考えられていなかった。
【００３７】
本発明は、このような状況に鑑みてなされたものであり、他のコネクションとの利用帯域の公平性を保持しつつ、利用可能帯域を十分に利用することができるようにするものである。
【課題を解決するための手段】
【００３８】
本発明の一側面の通信装置は、ネットワークを介した通信を行う通信装置において、直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測する予測手段と、予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信する送信手段とを備える。
【００３９】
本発明の一側面の通信装置において、前記予測手段は、前記直前の輻輳ウィンドウと、前記RTTの逆数の履歴を用いて、新たな輻輳ウィンドウを予測することができる。
【００４０】
本発明の一側面の通信装置において、前記予測手段は、前記直前の輻輳ウィンドウと、前記RTTの逆数の履歴を用いて、新たな輻輳ウィンドウを線形予測することができる。
【００４１】
本発明の一側面の通信方法は、ネットワークを介した通信を行う通信装置の通信方法において、直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測し、予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信するステップを含む。
【００４２】
本発明の一側面においては、直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウが予測され、予測された輻輳ウィンドウを用いて、ネットワークの帯域が割り当てられ、その帯域でデータが送信される。
【発明を実施するための最良の形態】
【００４３】
以下に本発明の実施の形態を説明するが、本発明の構成要件と、明細書又は図面に記載の実施の形態との対応関係を例示すると、次のようになる。この記載は、本発明をサポートする実施の形態が、明細書又は図面に記載されていることを確認するためのものである。従って、明細書又は図面中には記載されているが、本発明の構成要件に対応する実施の形態として、ここには記載されていない実施の形態があったとしても、そのことは、その実施の形態が、その構成要件に対応するものではないことを意味するものではない。逆に

、実施の形態が構成要件に対応するものとしてここに記載されていたとしても、そのことは、その実施の形態が、その構成要件以外の構成要件には対応しないものであることを意味するものでもない。

【００４４】

本発明の一側面の通信装置は、

ネットワークを介した通信を行う通信装置(例えば、図２の通信装置２０)において、

直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測する予測手段(例えば、図４の輻輳ウィンドウ決定部６３)と、

予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信する送信手段(例えば、図４の送信部６４)と

を備える。

【００４５】

本発明の一側面の通信方法は、

ネットワークを介した通信を行う通信装置(例えば、図２の通信装置２０)の通信方法において、

直前の輻輳ウィンドウとRTTの履歴を用いて、新たな輻輳ウィンドウを予測し(例えば、図５のステップＳ２２)、

予測された前記輻輳ウィンドウを用いて、前記ネットワークの帯域を割り当て、その帯域でデータを送信する（例えば、図５のステップＳ１２）

ステップを含む。

【００４６】

以下、本発明を適用した具体的な実施の形態について、図面を参照しながら詳細に説明する。

【００４７】

図２は、本発明を適用した通信装置の一実施の形態のハードウェアの構成例を示している。

【００４８】

図２の通信装置２０において、CPU（Central Processing Unit）２１，ROM（Read Only Memory）２２，RAM（Random Access Memory）２３は、バス２４により相互に接続されている。

【００４９】

バス２４には、さらに、入出力インターフェース２５が接続されている。入出力インターフェース２５には、キーボード、マウス、マイクロホン、リモートコントローラから送信されてくる指令を受信する受信部などよりなる入力部２６、ディスプレイ、スピーカなどよりなる出力部２７、ハードディスクや不揮発性のメモリなどよりなる記憶部２８、ネットワークインタフェースなどよりなる通信部２９、磁気ディスク、光ディスク、光磁気ディスク、或いは半導体メモリなどのリムーバブルメディア３１を駆動するドライブ３０が接続されている。

【００５０】

以上のように構成される通信装置２０では、CPU２１が、例えば、記憶部２８に記憶されているプログラムを、入出力インターフェース２５およびバス２４を介して、RAM２３にロードして実行することにより、記憶部２８などに記憶されているデータを、他の装置に送信する。

【００５１】

具体的には、CPU２１は、例えば、記憶部２８から送信対象とするデータを読み出し、通信部２９に供給する。通信部２９は、現行のTCPと互換性を有する、Saltというバージョンの TCPであるTCP Salt（後述する）にしたがって、CPU２１から供給されるデータを、インターネットを介して送信する。

【００５２】

通信装置２０のCPU２１が実行するプログラムは、例えば、リムーバブルメディア３１

に記録して、あるいは、ローカルエリアネットワーク、インターネット、デジタル衛星放送といった、有線または無線の伝送媒体を介して提供される。

【００５３】

そして、プログラムは、リムーバブルメディア３１をドライブ３０に装着することにより、入出力インターフェース２５を介して、記憶部２８にインストールすることができる。また、プログラムは、有線または無線の伝送媒体を介して、通信部２９で受信し、記憶部２８にインストールすることができる。その他、プログラムは、ROM２２や記憶部２８に、予めインストールしておくことができる。

【００５４】

次に、図３は、TCP Saltの状態遷移図である。

【００５５】

図３に示すように、TCP Saltでは、ブースト状態４１と輻輳回避状態４２の２つの状態がある。輻輳が検出されると、状態はブースト状態４１から輻輳回避状態４２に遷移する。また、利用可能帯域が増加して、利用可能帯域に余裕があると判断されると、状態は輻輳回避状態４２からブースト状態４１に遷移する。

【００５６】

ブースト状態４１では、正常なACKが受信されるたびに、輻輳ウィンドウcwndが１セグメント分増加する。一方、輻輳回避状態４２では、前回の輻輳ウィンドウcwndと、送信側で入手可能なRTTを用いて、以下の式（３）により、今回の輻輳ウィンドウとして最適な輻輳ウィンドウcwnd´が線形予測される。

【００５７】

【数３】

$$cwnd' = f(cwnd, IRTT_1, IRTT_2, \ldots, IRTT_n) = w_0 \times cwnd + \sum_{i=1}^{n} w_i IRTT_i \quad \cdots (3)$$

【００５８】

なお、式（３）において、$IRTT_i$はｉ（１≦ｉ≦ｎ）回前のRTTの逆数であり、w_0とw_iは重み係数である。式（３）によれば、輻輳ウィンドウcwnd´は、$IRTT_i$と前回の輻輳ウィンドウcwndが表す入力ベクトルと、重み係数w_0とw_iが表す係数ベクトルの内積、つまり、線形結合で表される。

【００５９】

即ち、IRTTは、輻輳の程度、即ち回線の混雑度に比例するものであるため、式（３）では、IRTTが回線の状況を表すものとして用いられる。そして、式（３）では、輻輳ウィンドウcwnd´は、過去の回線の状況を表すIRTTの履歴である$IRTT_i$と、その回線をどのように使用したかを表す直前の輻輳ウィンドウcwndとを用いた線形結合により予測される。

【００６０】

従って、式（３）では、過去の回線の状況を表す$IRTT_i$と、その回線をどのように使用したかを表す直前の輻輳ウィンドウcwndを用いて、現在の$IRTT_0$、つまり現在の回線の状況を予測し、それに応じた輻輳ウィンドウcwnd´を求めているといえる。

【００６１】

このように、TCP Saltでは、過去の回線の状況を表す$IRTT_i$と、その回線をどのように使用したかを表す直前の輻輳ウィンドウcwndを用いて、現在の回線の状況を予測するので、直前の回線の状況により現在の回線の状況を予測する場合に比べて、精度の高い予測を行うことができる。その結果、利用可能帯域を十分に利用することができる。

【００６２】

また、式（３）で逆数が用いられるRTTは、現行のTCPによる通信においても使用されるものであるため、式（３）により輻輳ウィンドウcwnd´を予測するTCP Saltは、従来のTCPとの互換性を有している。

【００６３】
以上のようにして式（３）により輻輳ウィンドウcwnd´が予測されると、その輻輳ウィンドウcwnd´を用いてインターネットの帯域が割り当てられ、その帯域でデータの送信が行われる。
【００６４】
なお、上述した重み係数w_0とw_iは、以下の式（４）にしたがって更新（機械学習）され、上述した輻輳ウィンドウcwnd´の予測に用いられる。
【００６５】
【数４】

$$w_0' = w_0 + C \times ERROR \times cwnd$$

$$w_i' = w_i + C \times ERROR \times IRTT_i \qquad \cdots (4)$$

【００６６】
なお、式（４）において、輻輳ウィンドウcwndは、式（３）において用いられる前回の輻輳ウィンドウcwndである。また、Ｃは所定の学習係数である。さらに、式（４）におけるERRORは、以下の式（５）により求められる。
【００６７】
【数５】

$$ERROR = \left(\sum_{j=0}^{n-1} \frac{RTT_{j+1} - RTT_j}{MAX\_RTT} \right) / n \qquad \cdots (5)$$

【００６８】
なお、式（５）において、RTT_jはｊ（０≦ｊ≦ｎ－１）回前のRTTであり、MAX_RTTは、RTTの最大値である。
【００６９】
上述した式（４）と式（５）によれば、輻輳の程度が減少する（$RTT_{j+1} > RTT_j$）と、重み係数w_0とw_iが増加し、輻輳の程度が増加する（$RTT_{j+1} < RTT_j$）と、重み係数w_0とw_iが減少する。また、輻輳の程度が不変（$RTT_{j+1} = RTT_j$）であれば、重み係数w_0とw_iも変化しない。その結果、重み係数w_0とw_iは、輻輳の程度が減少した場合、輻輳ウィンドウcwnd´を増加させ、輻輳の程度が増加した場合、輻輳ウィンドウcwnd´を減少させることができる。
【００７０】
従って、式（４）と式（５）によれば、回線を混雑させず、かつ、無駄にしない最適な輻輳ウィンドウcwnd´の予測に必要な重み係数w_0とw_iを、学習することができる。
【００７１】
以上のように、輻輳回避状態４２では、輻輳ウィンドウcwnd´の予測に用いられる重み係数w_0とw_iが更新され、その重み係数w_0とw_i、前回の輻輳ウィンドウcwnd、並びに過去ｎ回分の$IRTT_i$の履歴により、輻輳ウィンドウcwnd´が予測される。
【００７２】
なお、式（３）および式（４）では、$IRTT_i$そのものを用いるのではなく、以下の式（６）にしたがって正規化した$IRTT_i$を用いる。
【００７３】
【数６】

$$IRTT_i = \frac{1/RTT_i - 1/MAX\_RTT}{1/MIN\_RTT - 1/MAX\_RTT} \qquad \cdots (6)$$

【００７４】
なお、式（６）において、MIN_RTTは、RTTの最小値である。

【００７５】
図４は、図２の通信部２９の機能的構成例を示している。
【００７６】
図４の通信部２９は、RTT演算部６１、状態判定部６２、輻輳ウィンドウ決定部６３、送信部６４、および受信部６５により構成される。なお、通信部２９を構成する各ブロックは、必要に応じて相互に信号を授受することが可能とされている。
【００７７】
RTT演算部６１は、送信部６４から供給される、送信対象のデータ（以下、送信データという）に付加されたタイムスタンプを記憶する。また、RTT演算部６１は、記憶している送信データのタイムスタンプと、受信部６５から供給される、その送信データに対する確認応答であるACKに付加されたタイムスタンプとの差を、RTTとして演算する。RTT演算部６１は、そのRTTを状態判定部６２と輻輳ウィンドウ決定部６３に供給する。
【００７８】
状態判定部６２は、受信部６５から供給される輻輳の検出を表す情報に応じて、状態をブースト状態４１から輻輳回避状態４２に遷移させる。また、状態判定部６２は、RTT演算部６１から供給されるRTTの変動が０になったとき、状態を輻輳回避状態４２からブースト状態４１に遷移させる。即ち、利用可能帯域に余裕があることと、RTTの変動が少ないことに関連性があるため、状態判定部６２は、RTTの変動が０になったとき、利用可能帯域に余裕があると判断して、状態をブースト状態４１に遷移させる。また、状態判定部６２は、現在の状態を表す情報を輻輳ウィンドウ決定部６３に供給する。
【００７９】
輻輳ウィンドウ決定部６３は、RTT演算部６１から供給されるRTTを記憶する。また、輻輳ウィンドウ決定部６３は、状態判定部６２から供給される現在の状態を表す情報に応じて、輻輳ウィンドウcwndを決定する。具体的には、現在の状態がブースト状態４１である場合、輻輳ウィンドウ決定部６３は、RTT演算部６１からRTTが供給されるたびに、輻輳ウィンドウcwndを、前回の輻輳ウィンドウcwndから１セグメント分増加させて、今回の輻輳ウィンドウcwndとする。
【００８０】
また、現在の状態が輻輳回避状態４２である場合、輻輳ウィンドウ決定部６３は、過去ｎ回分と現在のRTT_j、予め設定されている学習係数Ｃ、前回の重み係数w_0とw_i、および前回の輻輳ウィンドウcwndを用いて、式（４）と式（５）にしたがって重み係数w_0とw_iを更新し、記憶する。
【００８１】
そして、輻輳ウィンドウ決定部６３は、記憶されている過去のｎ回分のRTT_i、更新された重み係数w_0およびw_i、並びに、前回の輻輳ウィンドウcwndを用いて、上述した式（３）にしたがって輻輳ウィンドウcwnd´を予測し、今回の輻輳ウィンドウcwndとする。輻輳ウィンドウ決定部６３は、今回の輻輳ウィンドウcwndを記憶するとともに、送信部６４に供給する。
【００８２】
送信部６４は、CPU２１（図２）から供給される送信データにセグメント単位でタイムスタンプを付加する。そして、送信部６４は、輻輳ウィンドウ決定部６３から供給される今回の輻輳ウィンドウcwndを用いて、インターネットの帯域を割り当て、その帯域でタイムスタンプが付加された送信データを送信する。また、送信部６４は、各セグメントに付加したタイムスタンプをRTT演算部６１に供給する。
【００８３】
受信部６５は、送信部６４により送信された送信データを受信する、通信装置２０とインターネットを介して接続された他の通信装置（図示せず）から、送信データに対するACKを受信する。そして、受信部６５は、受信したACKに付加されたタイムスタンプをRTT演算部６１に供給する。
【００８４】

また、受信部６５は、送信データが送信されてから予め設定された所定の時間内に、その送信データに対する、前に受信したACKと重複しない正常なACKが受信されたかを判定する。そして、所定の時間内に正常なACKが受信されていないと判定された場合、受信部６５は、輻輳を検出し、輻輳の検出を表す情報を状態判定部６２に供給する。一方、所定の時間内に正常なACKが受信されたと判定された場合、受信部６５は輻輳を検出しない。

【００８５】

次に、図５のフローチャートを参照して、図４の通信部２９による、輻輳ウィンドウcwndを決定する決定処理について説明する。この決定処理は、例えば、図２のCPU２１から送信部６４に送信データが供給されたとき、開始される。

【００８６】

ステップＳ１１において、送信部６４は、CPU２１から供給された送信データに、セグメント単位でタイムスタンプを付加し、そのタイムスタンプをRTT演算部６１に供給する。ステップＳ１２において、送信部６４は、輻輳ウィンドウ決定部６３により決定された今回の輻輳ウィンドウcwndを用いて、インターネットの帯域を割り当て、その帯域でタイムスタンプを付加した送信データを他の通信装置に送信する。

【００８７】

ステップＳ１３において、受信部６５は、ステップＳ１２で送信された送信データに対するACKが受信されたかどうかを判定する。ステップＳ１３でACKが受信されていないと判定された場合、ステップＳ１４において、受信部６５は、予め設定された所定の時間が経過したかどうかを判定する。ステップＳ１４で所定の時間が経過していないと判定された場合、処理はステップＳ１３に戻り、ACKが受信されるか、または、所定の時間が経過するまで、ステップＳ１３およびＳ１４の処理が繰り返される。

【００８８】

一方、ステップＳ１４で所定の時間が経過したと判定された場合、即ち、タイムアウトが発生した場合、受信部６５は、輻輳を検出して、輻輳の検出を表す情報を状態判定部６２に供給する。そして、処理はステップＳ２０に進む。

【００８９】

また、ステップＳ１３でACKが受信されたと判定された場合、ステップＳ１５において、受信部６５は、受信したACKが正常なACKであるかどうか、即ち、前回受信したACKと重複しないACKであるかどうかを判定する。ステップＳ１５で、受信したACKが正常なACKではないと判定された場合、受信部６５は、輻輳を検出して、輻輳の検出を表す情報を状態判定部６２に供給する。そして、処理はステップＳ２０に進む。

【００９０】

一方、ステップＳ１５で、受信したACKが正常なACKであると判定された場合、受信部６５は、輻輳を検出せず、受信したACKに付加されているタイムスタンプをRTT演算部６１に供給する。そして、処理はステップＳ１６に進む。ステップＳ１６において、RTT演算部６１は、ステップＳ１１で送信部６４から供給されたタイムスタンプと、ステップＳ１５で受信部６５から供給されたタイムスタンプとの差をRTTとして演算する。そして、RTT演算部６１は、そのRTTを状態判定部６２と輻輳ウィンドウ決定部６３に供給する。

【００９１】

ステップＳ１７において、状態判定部６２は、RTTの変動がないかどうか、即ち、RTT演算部６１から供給された最新のRTTと、１つ前にRTT演算部６１から供給されたRTTとの差がないかどうかを判定する。ステップＳ１７でRTTの変動がないと判定された場合、ステップＳ１８において、状態判定部６２は、状態をブースト状態４１にする。そして、状態判定部６２は、現在の状態がブースト状態４１であることを表す情報を、輻輳ウィンドウ決定部６３に供給する。

【００９２】

ステップＳ１９において、輻輳ウィンドウ決定部６３は、状態判定部６２から供給される、現在の状態がブースト状態であることを表す情報に応じて、輻輳ウィンドウcwndを１セグメント分増加させる。そして、輻輳ウィンドウ決定部６３は、その輻輳ウィンドウcw

ndを、今回の輻輳ウィンドウcwndとして記憶するとともに、送信部６４に供給する。この今回の輻輳ウィンドウcwndは、次回のステップＳ１２の処理で用いられる。

【００９３】

一方、ステップＳ１７でRTTの変動があると判定された場合、処理はステップＳ２０に進む。ステップＳ２０において、状態判定部６２は、状態を輻輳回避状態４２にする。そして、状態判定部６２は、現在の状態が輻輳回避状態４２であることを表す情報を、輻輳ウィンドウ決定部６３に供給する。

【００９４】

ステップＳ２１において、輻輳ウィンドウ決定部６３は、現在の状態が輻輳回避状態４２であることを表す情報に応じて、過去ｎ回分と現在のRTT_j、予め設定されている学習係数Ｃ、前回の重み係数w_0とw_i、および前回の輻輳ウィンドウcwndを用いて、上述した式（４）と式（５）にしたがって重み係数w_0とw_1を更新し、記憶する。

【００９５】

ステップＳ２２において、輻輳ウィンドウ決定部６３は、現在の状態が輻輳回避状態４２であることを表す情報に応じて、記憶されている過去のｎ回分のRTT_i、ステップＳ２１で更新された重み係数w_0およびw_i、並びに、前回の輻輳ウィンドウcwndを用いて、上述した式（３）にしたがって輻輳ウィンドウcwnd´を予測する。そして、輻輳ウィンドウ決定部６３は、その輻輳ウィンドウcwnd´を、今回の輻輳ウィンドウcwndとして記憶するとともに、送信部６４に供給する。この今回の輻輳ウィンドウcwndは、次回のステップＳ１２の処理で用いられる。

【００９６】

ステップＳ１９またはＳ２２の処理後、ステップＳ２３において、送信部６４は、CPU２１から供給された全ての送信データを送信したかどうかを判定する。ステップＳ２３で、まだ全ての送信データを送信していないと判定された場合、処理はステップＳ１１に戻り、上述した処理が繰り返される。一方、ステップＳ２３で、全ての送信データを送信したと判定された場合、処理は終了する。

【００９７】

以下に、上述したTCP Saltによる通信の動作について、シミュレーション結果を用いて説明する。このシミュレーションは、TCP Saltにおける学習係数Ｃを0.6とし、RTTの履歴の数であるｎを5として、ネットワークシミュレータNS-2を用いて行われる。

【００９８】

まず最初に、第１のシミュレーションについて説明する。

【００９９】

図６は、第１のシミュレーションのネットワークトポロジを示している。

【０１００】

図６のネットワークトポロジでは、２つの中間ノード８１と８２を挟んで、２つの別々のノードであるノード８０およびノード８４と、２つの別々のノードであるノード８３およびノード８５が接続されている。そして、UDP（User Datagram Protocol）に準拠してデータが、ノード８０から中間ノード８１と８２を通ってノード８３へ流され、TCP SaltまたはTCP Renoに準拠してデータが、ノード８４から中間ノード８１と８２を通ってノード８５へ流される。

【０１０１】

なお、TCP SaltとTCP Renoのデータフローに対する受信端末はTCP Sinkであり、TCP SaltおよびTCP Renoの最大のウィンドウサイズは３３セグメントである。このことは、後述するシミュレーションでも同様である。

【０１０２】

また、図６のネットワークトポロジでは、全帯域が100Mbpsとなっており、ノード８０から中間ノード８１への流れ、ノード８４から中間ノード８１への流れ、中間ノード８２からノード８３への流れ、および中間ノード８２からノード８５への流れによって、データに5msの遅延が発生する。さらに、中間ノード８１から中間ノード８２への流れによっ

て、データに40msの遅延が発生する。

【０１０３】

第１のシミュレーションでは、100Mbpsの全帯域のうち、UDPに準拠したデータフロー（以下、UDPフローという）によって98.5Mbpsの帯域を圧迫した状態で、TCP SaltまたはTCP Renoに準拠してデータが流される。詳細には、第１のシミュレーションが開始されてから4秒後にUDPフローが開始され、5秒後にTCP SaltまたはTCP Renoに準拠したデータフロー（以下、TCP SaltフローまたはTCP Renoフローという）が開始される。そして、50秒後にTCP SaltフローまたはTCP Renoフローが、51秒後にUDPフローが、55秒後に第１のシミュレーション自体が、停止される。

【０１０４】

次に、図７乃至図９を参照して、第１のシミュレーションの結果について説明する。

【０１０５】

図７は、第１のシミュレーションのTCP SaltフローとTCP Renoフローにおける、輻輳ウィンドウcwndの時間変化を示すグラフである。なお、図７において、横軸は第１のシミュレーションを開始してからの時間(sec)を表し、縦軸は輻輳ウィンドウcwnd(セグメント)を表している。

【０１０６】

図７に示すように、TCP Saltフローでは、ブースト状態４１から輻輳回避状態４２に最初に遷移したときに輻輳ウィンドウcwndが減少し、それ以降、比較的大きい値に徐々に収束していく。即ち、輻輳回避状態４２に最初に遷移したときには、まだ重み係数w_0およびw_iが学習されていないため、TCP Saltフローでは、輻輳ウィンドウcwndを正確に予測することができないが、その後、徐々に重み係数w_0およびw_iが学習されていくので、その重み係数w_0およびw_iを用いて、輻輳ウィンドウcwndを正確に予測することが可能となる。その結果、輻輳ウィンドウcwndは徐々に収束していく。

【０１０７】

これに対して、TCP Renoフローでは、図７に示すように、スロースタート状態１１から輻輳回避状態１２に最初に遷移した後も、一定周期で輻輳ウィンドウcwndが減少する。即ち、TCP Renoフローでは、輻輳ウィンドウcwndを減少させることができないため、UDPフローによる圧迫で輻輳が発生し、その輻輳が検出されるたびに、輻輳ウィンドウcwndが減少する。

【０１０８】

図８は、第１のシミュレーションのTCP SaltフローとTCP Renoフローにおける、スループットの時間変化を示すグラフである。なお、図８において、横軸は第１のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１０９】

図８に示すように、TCP Saltフローでは、ブースト状態４１から輻輳回避状態４２に最初に遷移したときにのみスループットが低下するだけで、それ以降は、スループットが、利用可能帯域の最大限の値である、1.5Mbps付近で安定する。これにより、TCP Saltフローでは、最適な輻輳ウィンドウcwndを予測することができていることがわかる。

【０１１０】

これに対して、図８に示すように、TCP Renoフローでは、スロースタート状態１１から輻輳回避状態１２に最初に遷移した後も、何回もスループットが低下している。即ち、図７で示したように、TCP Renoフローでは、UDPフローの圧迫によって輻輳が発生するたびに、輻輳ウィンドウcwndが低下するので、スループットも低下する。

【０１１１】

図９は、第１のシミュレーションのTCP SaltフローとTCP Renoフローにおける、平均スループット、最大スループット、および転送データ量を表している。

【０１１２】

図９に示すように、TCP Renoフローの平均スループットは、1.084Mbpsであり、TCP Saltフローの平均スループットは、1.428である。従って、TCP Renoフローに対するTCP Salt

フローの平均スループットの改善率は31.7％である。

【０１１３】

また、図９に示すように、TCP Renoフローの最大スループットは、2.030Mbpsであり、TCP Saltフローの最大スループットは、TCP Renoフローの場合と同一の2.030Mbpsである。従って、TCP Renoフローに対するTCP Saltフローの最大スループットの改善率は0.0％である。

【０１１４】

さらに、図９に示すように、TCP Renoフローの転送データ量は、6.096Mbであり、TCP Saltフローの転送データ量は、8.030Mbである。従って、TCP Renoフローに対するTCP Saltフローの転送データ量の改善率は31.7％である。

【０１１５】

以上により、第１のシミュレーションでは、TCP Saltフローにおける平均スループットと転送データ量が、TCP Renoフローに比べて改善されることがわかる。

【０１１６】

次に、第２のシミュレーションについて説明する。

【０１１７】

第２のシミュレーションのネットワークトポロジは、図６に示した第１のシミュレーションのネットワークトポロジと同一であるので、説明は省略する。なお、第２のシミュレーションでは、第１のシミュレーションと異なり、UDPフローにより圧迫される帯域が常に98.5Mbpsであるのではなく、第２のシミュレーションが開始されてから20秒後から35秒後までの15秒間、UDPフローにより圧迫される帯域が50Mbpsに低下する。

【０１１８】

図１０は、第２のシミュレーションのTCP SaltフローとTCP Renoフローにおける、スループットの時間変化について説明する。なお、図１０において、横軸は第２のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１１９】

図１０に示すように、TCP Saltフローでは、TCP Renoフローに比べて、圧迫された帯域が低下する、第２のシミュレーションを開始してから20秒後に、即座にスループットが向上する。これは、UDPフローにより圧迫される帯域が低下し、TCP Saltフローの利用可能帯域が増加することによって、RTTの変動がなくなるため、TCP Saltフローでは、即座に状態が輻輳回避状態４２からブースト状態４１に遷移し、輻輳ウィンドウcwndの増加速度が大きくなるためである。

【０１２０】

これに対して、TCP Renoフローでは、状態が輻輳回避状態１２のままであるため、輻輳ウィンドウcwndの増加速度が、TCP Saltフローに比べて小さい。

【０１２１】

以上のように、TCP Saltフローでは、RTTの変動により、現在の帯域の状態が把握されるので、TCP Renoフローと比較して、圧迫される帯域の変化に迅速に反応することができる。

【０１２２】

また、TCP SaltフローとTCP Renoフローでは、圧迫された帯域が低下している第２のシミュレーションを開始してから20秒後から35秒までの15秒間、スループットは2.75Mbps付近で安定する。このとき、TCP SaltフローとTCP Renoフローのスループットは、利用可能な50Mbpsの帯域よりも小さいので、輻輳は発生しない。

【０１２３】

なお、第２のシミュレーションが開始されてから20秒後から35秒後までの15秒間ではなく、20秒後付近でTCP Renoフローのスループットが一番低下している18秒後から、33秒後までの15秒間、UDPフローにより圧迫される帯域が50Mbpsに低下すると、図１１に示すように、圧迫される帯域の変化への反応の速さの差は、さらに顕著になる。

【０１２４】
次に、第３のシミュレーションについて説明する。
【０１２５】
図１２は、第３のシミュレーションのネットワークトポロジを示している。
【０１２６】
図１２のネットワークトポロジでは、２つの中間ノード１０１と１０２を挟んで、２つの別々のノードであるノード１００およびノード１０４と、２つの別々のノードであるノード１０３およびノード１０５が接続されている。そして、第１のTCPのいずれかのバージョンに準拠したフロー（以下、TCPフローという）として、TCP SaltまたはTCP Renoに準拠して、ノード１００から中間ノード１０１と１０２を通ってノード１０３へデータが流され、第２のTCPフローとして、TCP SaltフローまたはTCP Renoフローが、ノード１０４から中間ノード１０１と１０２を通ってノード１０５へ流される。
【０１２７】
なお、図１２のネットワークトポロジでは、中間ノード１０１と１０２間の帯域が2.5Mbpsとなっており、それ以外の帯域が100Mbpsとなっている。また、ノード１００から中間ノード１０１への流れ、ノード１０４から中間ノード１０１への流れ、中間ノード１０２からノード１０３への流れ、および中間ノード１０２からノード１０５への流れによって、データに5msの遅延が発生する。さらに、中間ノード１０１から中間ノード１０２への流れによって、データに40msの遅延が発生する。
【０１２８】
第３のシミュレーションでは、第１のTCPフローおよび第２のTCPフローが、TCP Saltフローである。詳細には、第１のシミュレーションが開始されてから1秒後に第１のTCPフローとしての第１のTCP Saltフローが開始され、5秒後に第２のTCPフローとしての第２のTCP Saltフローが開始される。そして、45秒後に第１のTCP Saltフローが、50秒後に第２のTCP Saltフローが、55秒後に第３のシミュレーション自体が、停止される。
【０１２９】
図１３は、第３のシミュレーションの第１および第２のTCP Saltフローにおける、スループットの時間変化を示すグラフである。なお、図１３において、横軸は第３のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。
【０１３０】
図１３に示すように、第１のTCP Saltフローが開始される1秒後から、第２のTCP Saltフローが開始される5秒後までの間、第１のTCPフローでは、スループットが2.5Mbpsとなっており、利用可能帯域が最大限利用されている。
【０１３１】
また、同様に、第１のTCP Saltフローが停止される45秒後から、第２のTCP Saltフローが停止される50秒後までの間、第２のTCP Saltフローでは、スループットが2.5Mbpsとなっており、利用可能帯域が最大限利用されている。
【０１３２】
さらに、第１と第２のTCP Saltフローが両方行われている、5秒後から45秒後までの間、図１３に示すように、多少の変動はあるものの、利用可能帯域は公平に利用されている。即ち、TCP Saltフローどうしが帯域を共有した場合、利用可能帯域が公平に利用される。
【０１３３】
次に、第４のシミュレーションについて説明する。
【０１３４】
第４のシミュレーションのネットワークトポロジは、図１２に示した第３のシミュレーションのネットワークトポロジと同一であるので、説明は省略する。なお、第４のシミュレーションでは、第３のシミュレーションと異なり、第２のTCPフローとして、TCP Renoに準拠してデータが流される。即ち、第４のシミュレーションでは、第１のTCPフローが

、TCP Saltフローであり、第２のTCPフローが、TCP Renoフローである。

【０１３５】

図１４は、第４のシミュレーションのTCP SaltフローおよびTCP Renoフローにおける、スループットの時間変化を示すグラフである。なお、図１４において、横軸は第４のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１３６】

図１４に示すように、TCP Saltフローが開始される1秒後から、TCP Renoフローが開始される5秒後までの間、TCP Saltフローでは、スループットが2.5Mbpsとなっており、利用可能帯域が最大限利用されている。

【０１３７】

また、同様に、TCP Saltフローが停止される45秒後から、TCP Renoフローが停止される50秒後までの間、TCP Renoフローでは、スループットが2.5Mbpsとなっており、利用可能帯域が最大限利用されている。

【０１３８】

さらに、TCP SaltフローとTCP Renoフローが両方行われている、5秒後から45秒後までの間、図１４に示すように、多少の変動はあるものの、利用可能帯域は公平に利用されている。即ち、TCP SaltフローとTCP Renoフローが帯域を共有した場合にも、利用可能帯域が公平に利用される。

【０１３９】

以上の第１乃至第４のシミュレーションにより、TCP Saltフローでは、他のコネクションとの利用帯域の公平性を保持しつつ、利用可能帯域を十分に利用することができることがわかる。また、TCP Saltフローでは、利用可能帯域の変化に迅速に対応することができることがわかる。

【０１４０】

次に、TCP Saltにおいて輻輳ウィンドウcwnd´の予測時に用いられる重み係数w_0とw_iの推移について、シミュレーション結果を用いて説明する。

【０１４１】

まず最初に、図１５を参照して、第１のシミュレーション中の重み係数w_0乃至w_5の推移について説明する。なお、図１５において、横軸は、第１のシミュレーションを開始してからの時間（sec）を表し、縦軸は重み係数w_0乃至w_5の値を表している。このことは、後述する図１６においても同様である。

【０１４２】

図１５に示すように、第１のシミュレーションにおいて、重み係数w_0乃至w_5は、一定時間で収束していく。また、第１のシミュレーションでは、直近の過去の$IRTT_1$に対する重み係数w_1が一番大きく、前回の輻輳ウィンドウcwndに対する重み係数w_0はあまり大きくない。

【０１４３】

次に、図１６は、第２のシミュレーション中の重み係数w_0乃至w_5の推移を示している。なお、図１６において、第２のシミュレーションは、第２のシミュレーションが開始されてから20秒後から35秒後までの15秒間、UDPフローにより圧迫される帯域が50Mbpsに低下するものである。

【０１４４】

上述したように、第２のシミュレーションにおいて、UDPフローにより圧迫される帯域が50Mbpsに低下する20秒後から35秒後までの15秒間、輻輳は発生せず、TCP Saltフローでは、ブースト状態４１が維持されるので、図１６に示すように、重み係数w_0乃至w_5は変化しない。また、第２のシミュレーションでは、第１のシミュレーションと同様に、直近の過去の$IRTT_1$に対する重み係数w_1が一番大きく、前回の輻輳ウィンドウcwndに対する重み係数w_0はあまり大きくない。

【０１４５】

以上のように、輻輳ウィンドウcwndに対する重み係数w_0は、あまり大きくない。従って、以下に、輻輳ウィンドウcwnd´の予測時に用いられる前回の輻輳ウィンドウcwndの必要性について、シミュレーション結果を用いて説明する。

【０１４６】

まず最初に、第５のシミュレーションについて説明する。

【０１４７】

第５のシミュレーションは、第１のシミュレーションにおいて、TCP Saltフローまたは TCP Renoフローではなく、TCP Saltの第１または第２の変形に準拠したフロー(以下、第１変形TCP Saltフローまたは第２変形TCP Saltフローという)もしくはTCP Saltフローを用いたものである。

【０１４８】

ここで、TCP Saltの第１の変形とは、TCP Saltにおいて重み係数w_0を０とし、RTTの履歴の数であるｎを５としたものであり、第２の変形とは、TCP Saltにおいて重み係数w_0を０とし、ｎを６としたものである。

【０１４９】

図１７は、第５のシミュレーションの第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローにおける、スループットの時間変化を示すグラフである。なお、図１７において、横軸は第５のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１５０】

図１７に示すように、第５のシミュレーションでは、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローにおけるスループットの差はない。

【０１５１】

次に、第６のシミュレーションについて説明する。

【０１５２】

第６のシミュレーションは、第２のシミュレーションにおいて、TCP Saltフローまたは TCP Renoフローではなく、第１変形TCP Saltフロー、第２変形TCP Saltフロー、またはTCP Saltフローを用いたものである。

【０１５３】

図１８は、第６のシミュレーションの第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローにおける、スループットの時間変化を示すグラフである。なお、図１８において、横軸は第６のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１５４】

図１８に示すように、第６のシミュレーションにおいても、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローにおけるスループットの差はない。

【０１５５】

次に、第７のシミュレーションについて説明する。

【０１５６】

図１９は、第７のシミュレーションのネットワークトポロジを示している。

【０１５７】

図１９のネットワークトポロジでは、２つの中間ノード１２１と１２２を挟んで、３つの別々のノードであるノード１２０、ノード１２４、およびノード１２６と、３つの別々のノードであるノード１２３、ノード１２５、およびノード１２７が接続されている。

【０１５８】

また、UDPフローとして、UDPに準拠して、ノード１２０から中間ノード１２１と１２２を通ってノード１２３へデータが流され、第１のTCPフローとして、TCP Reno、TCP Vegas、TCP Saltの第１変形、TCP Saltの第２変形、またはTCP Saltに準拠して、ノード１２４から中間ノード１２１と１２２を通ってノード１２５へデータが流される。さらに、第２のTCPフローとして、TCP Reno、TCP Vegas、TCP Saltの第１変形、TCP Saltの第２変形、

またはTCP Saltに準拠して、ノード１２６から中間ノード１２１と１２２を通ってノード１２７へデータが流される。

【０１５９】

なお、図１９のネットワークトポロジでは、全帯域が100Mbpsとなっている。また、ノード１２０から中間ノード１２１への流れ、ノード１２４から中間ノード１２１への流れ、ノード１２６から中間ノード１２１への流れ、中間ノード１２２からノード１２３への流れ、中間ノード１２２からノード１２５への流れ、および中間ノード１２２からノード１２７への流れによって、データに5msの遅延が発生する。さらに、中間ノード１２１から中間ノード１２２への流れによって、データに40msの遅延が発生する。

【０１６０】

第７のシミュレーションでは、第１のTCPフローおよび第２のTCPフローが、TCP Renoフローである。

【０１６１】

図２０は、第７のシミュレーションの第１および第２のTCP Renoフローにおける、スループットの時間変化を示すグラフである。なお、図２０において、横軸は第７のシミュレーションを開始してからの時間（sec）を表し、縦軸はスループット（Mbps）を表している。

【０１６２】

図２０に示すように、第７のシミュレーションでは、第１のTCP Renoフローにおけるスループットを表す実線と、第２のTCP Renoフローにおけるスループットを表す点線があまり重なっていない。即ち、TCP Renoフローどうしが帯域を共有した場合、利用可能帯域は公平に利用されない。

【０１６３】

次に、第８乃至第１５のシミュレーションについて説明する。

【０１６４】

第８乃至第１５のシミュレーションのネットワークトポロジは、図１９に示した第７のシミュレーションのネットワークトポロジと同一であるので、説明は省略する。なお、第８のシミュレーションでは、第７のシミュレーションと異なり、第１および第２のTCPフローが、TCP Vegasに準拠したデータフロー(以下、TCP Vegasフローという)である。

【０１６５】

図２１は、第８のシミュレーションの第１および第２のTCP Vegasフローにおける、スループットの時間変化を示すグラフである。

【０１６６】

図２１に示すように、第８のシミュレーションでは、第１のTCP Vegasフローと第２のTCP Vegasフローのうち、最初に開始した方が帯域の利用において優位になる。即ち、TCP Vegasフローどうしが帯域を共有した場合、利用可能帯域は公平に利用されない。

【０１６７】

次に、第９乃至第１１のシミュレーションについて説明する。

【０１６８】

第９のシミュレーションでは、第１および第２のTCPフローが、第１変形TCP Saltフローであり、第１０のシミュレーションでは、第２変形TCP Saltフロー、第１１のシミュレーションでは、TCP Saltフローである。

【０１６９】

図２２乃至図２４は、第９乃至第１１のシミュレーションの第１および第２の第１変形TCP Saltフロー、第１および第２の第２変形TCP Saltフロー、並びに第１および第２のTCP Saltフローのそれぞれにおける、スループットの時間変化を示すグラフである。

【０１７０】

図２２乃至図２４に示すように、第１１のシミュレーション、第９のシミュレーション、第１０のシミュレーションの順に、帯域利用の公平性が高い。即ち、TCP Saltフローどうしが帯域を共有した場合が最も公平性が高く、第１変形TCP Saltフローどうしが帯域を

共有した場合が、その次に公平性が高く、第２変形TCP Saltフローどうしが帯域を共有した場合が、最も公平性が低い。

【０１７１】

次に、第１２乃至第１４のシミュレーションについて説明する。

【０１７２】

第１２のシミュレーションでは、第１のTCPフローが、第１変形TCP Saltフローであり、第１３のシミュレーションでは、第２変形TCP Saltフローであり、第１４のシミュレーションでは、TCP Saltフローである。また、第１２乃至第１４のシミュレーションでは、第２のTCPフローが、TCP Renoフローである。

【０１７３】

図２５乃至図２７は、第１２乃至第１４のシミュレーションの、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローのそれぞれと、TCP Renoフローにおける、スループットの時間変化を示すグラフである。

【０１７４】

図２５に示すように、第１２のシミュレーションでは、第１変形TCP Saltフローの方が、TCP Renoフローに比べて、帯域利用において優位になっている。また、図２６に示すように、第１３のシミュレーションでは、第１変形TCP Saltフローと同様に、第２変形TCP Saltフローの方が、TCP Renoフローに比べて、帯域利用において優位になっている。さらに、図２７に示すように、第１４のシミュレーションでは、TCP SaltフローとTCP Renoフローの利用帯域がほぼ公平になっている。

【０１７５】

次に、第１５のシミュレーションについて説明する。

【０１７６】

第１５のシミュレーションでは、第１のTCPフローが、TCP Vegasフローであり、第２のTCPフローが、TCP Renoフローである。

【０１７７】

図２８は、第１５のシミュレーションのTCP VegasフローとTCP Renoフローにおける、スループットの時間変化を示すグラフである。

【０１７８】

図２８に示すように、第１５のシミュレーションでは、TCP VegasフローとTCP Renoフローの利用帯域の公平性は比較的良いが、TCP VegasフローとTCP Renoフローにおけるスループットが小さい。

【０１７９】

図２９は、第７、第８、および第１５のシミュレーションにおける転送量（byte）を示す表であり、図３０は、第９乃至第１１のシミュレーションにおける転送量（byte）を示す表である。また、図３１は、第１２乃至第１４のシミュレーションにおける転送量（byte）を示す表である。さらに、図３２は、第７乃至第１５のシミュレーションにおける転送量（byte）を表すグラフである。

【０１８０】

図２９に示すように、第７のシミュレーションでは、第１のTCPフローとしてのTCP Renoフローにおける転送量が3410バイトであり、第２のTCPフローとしてのTCP Renoフローにおける転送量が4104バイトである。従って、図２９や図３２に示すように、第７のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が7514バイトであり、差の絶対値が694である。

【０１８１】

第８のシミュレーションでは、図２９に示すように、第１のTCPフローとしてのTCP Vegasフローにおける転送量が7851バイトであり、第２のTCPフローとしてのTCP Vegasフローにおける転送量が2176バイトである。従って、図２９や図３２に示すように、第８のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が10027バイトであり、差の絶対値が5675である。

【０１８２】
第１５のシミュレーションでは、図２９に示すように、第１のTCPフローとしてのTCP Vegasフローにおける転送量が3320バイトであり、第２のTCPフローとしてのTCP Renoフローにおける転送量が3700バイトである。従って、図２９や図３２に示すように、第１５のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が7020バイトであり、差の絶対値が380である。
【０１８３】
また、図３０に示すように、第９のシミュレーションでは、第１のTCPフローとしての第１の第１変形TCP Saltフローにおける転送量が3849バイトであり、第２のTCPフローとしての第２の第１変形TCP Saltフローにおける転送量が4397バイトである。従って、図３０や図３２に示すように、第９のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8246バイトであり、差の絶対値が548である。
【０１８４】
第１０のシミュレーションでは、図３０に示すように、第１のTCPフローとしての第１の第２変形TCP Saltフローにおける転送量が3818バイトであり、第２のTCPフローとしての第２の第２変形TCP Saltフローにおける転送量が4462バイトである。従って、図３０や図３２に示すように、第１０のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8280バイトであり、差の絶対値が644である。
【０１８５】
第１１のシミュレーションでは、図３０に示すように、第１のTCPフローとしての第１のTCP Saltフローにおける転送量が3960バイトであり、第２のTCPフローとしての第２のTCP Saltフローにおける転送量が4273バイトである。従って、図３０や図３２に示すように、第１１のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8233バイトであり、差の絶対値が313である。
【０１８６】
さらに、図３１に示すように、第１２のシミュレーションでは、第１のTCPフローとしての第１変形TCP Saltフローにおける転送量が4743バイトであり、第２のTCPフローとしてのTCP Renoフローにおける転送量が3352バイトである。従って、図３１や図３２に示すように、第１２のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8095バイトであり、差の絶対値が1391である。
【０１８７】
第１３のシミュレーションでは、図３１に示すように、第１のTCPフローとしての第２変形TCP Saltフローにおける転送量が4783バイトであり、第２のTCPフローとしてのTCP Renoフローにおける転送量が3456バイトである。従って、図３１や図３２に示すように、第１３のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8239バイトであり、差の絶対値が1327である。
【０１８８】
第１４のシミュレーションでは、図３１に示すように、第１のTCPフローとしてのTCP Saltフローにおける転送量が4316バイトであり、第２のTCPフローとしてのTCP Renoフローにおける転送量が3824バイトである。従って、図３１や図３２に示すように、第１４のシミュレーションでは、第１のTCPフローと第２のTCPフローにおける転送量の和が8140バイトであり、差の絶対値が492である。
【０１８９】
図２９乃至図３２に示すように、第８のシミュレーションでは、第１のTCPフローとしてのTCP Vegasフローにおける転送量と、第２のTCPフローとしてのTCP Vegasフローにおける転送量の和が、最大となっているが、それらの転送量の差も最大となっている。即ち、TCP Vegasフローどうしが帯域を共有した場合、全体的な転送量は多いが、公平性が悪い。
【０１９０】
また、第７や第１５のシミュレーションでは、第１のTCPフローとしてのTCP Renoフロ

ーまたはTCP Vegasフローにおける転送量と、第２のTCPフローとしてのTCP Renoフローにおける転送量の差は小さいが、それらの転送量の和も小さくなっている。即ち、TCP Renoフローどうしや、TCP RenoフローとTCP Vegasフローが帯域を共有した場合、公平性は比較的良いが、全体的な転送量は少なくなる。

【０１９１】

これに対して、第９乃至第１４のシミュレーションでは、第１のTCPフローとしての第１変形TCP Saltフロー、第２変形TCP Saltフロー、またはTCP Saltフローにおける転送量と、第２のTCPフローとしての第１変形TCP Saltフロー、第２変形TCP Saltフロー、TCP Saltフロー、またはTCP Renoフローにおける転送量の差は小さいが、それらの転送量の和は比較的大きい。

【０１９２】

即ち、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローどうしや、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローのそれぞれとの TCP Renoフローが帯域を共有した場合、利用帯域の公平性が良く、かつ、全体的な転送量が多い。

【０１９３】

また、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローどうしが帯域を共有した場合と、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローのそれぞれと TCP Renoフローが帯域を共有した場合とで、全体的な転送量には、ほぼ差がないので、現行のTCP Renoを一度にTCP Saltに移行せず、徐々に移行する場合であっても、問題がないことがわかる。

【０１９４】

さらに、第９乃至第１４のシミュレーションにおいて、第１のTCPフローと第２のTCPのフローにおける転送量の和はほぼ同一であるが、差については差がある。具体的には、第９と第１０のシミュレーションに比べて、第１１のシミュレーションにおける差が小さく、第１２と第１３のシミュレーションに比べて、第１４のシミュレーションにおける差が小さい。即ち、第１変形TCP Saltフロー、第２変形TCP Saltフロー、およびTCP Saltフローの中で、TCP Saltフローが最も公平性が良い。従って、重み係数w_0は、利用帯域の公平性を保持するために必要であることがわかる。

【０１９５】

なお、マルチメディアデータを配信する場合、通信装置２０は、TCP Saltにしたがって決定された輻輳ウィンドウcwndに基づいて、インターネットの帯域を割り当てるとともに、特許文献３に記載されている制御方法で、インターネットの状態に応じて個々の送信データの送信間隔を決定し、その送信間隔で送信データを、割り当てられた帯域を用いて送信するようにしてもよい。

【０１９６】

また、本明細書において、プログラム記録媒体に格納されるプログラムを記述するステップは、記載された順序に沿って時系列的に行われる処理はもちろん、必ずしも時系列的に処理されなくとも、並列的あるいは個別に実行される処理をも含むものである。

【０１９７】

さらに、本発明の実施の形態は、上述した実施の形態に限定されるものではなく、本発明の要旨を逸脱しない範囲において種々の変更が可能である。

【図面の簡単な説明】

【０１９８】

【図１】TCP Renoの状態遷移図である。

【図２】本発明を適用した通信装置の一実施の形態のハードウェアの構成例を示す図である。

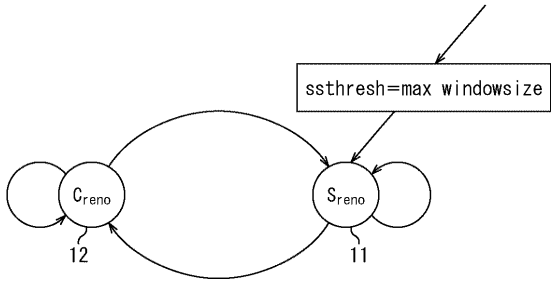
【図３】TCP Saltの状態遷移図である。

【図４】図２の通信部２９の機能的構成例を示す図である。

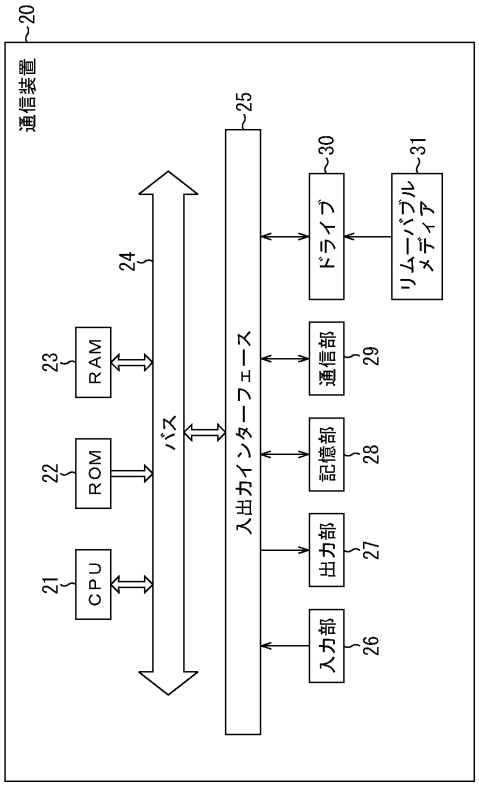
【図５】通信部による決定処理について説明するフローチャートである。

【図６】第１のシミュレーションのネットワークトポロジを示す図である。
【図７】第１のシミュレーションの輻輳ウィンドウの時間変化を示すグラフである。
【図８】第１のシミュレーションのスループットの時間変化を示すグラフである。
【図９】第１のシミュレーションの平均スループット、最大スループット、および転送データ量を表す図である。
【図１０】第２のシミュレーションのスループットの時間変化を示すグラフである。
【図１１】他の第２のシミュレーションのスループットの時間変化を示すグラフである。
【図１２】第３のシミュレーションのネットワークトポロジを示す図である。
【図１３】第３のシミュレーションのスループットの時間変化を示すグラフである。
【図１４】第４のシミュレーションのスループットの時間変化を示すグラフである。
【図１５】第１のシミュレーション中の重み係数の推移を示すグラフである。
【図１６】第２のシミュレーション中の重み係数の推移を示すグラフである。
【図１７】第５のシミュレーションのスループットの時間変化を示すグラフである。
【図１８】第６のシミュレーションのスループットの時間変化を示すグラフである。
【図１９】第７のシミュレーションのネットワークトポロジを示す図である。
【図２０】第７のシミュレーションのスループットの時間変化を示すグラフである。
【図２１】第８のシミュレーションのスループットの時間変化を示すグラフである。
【図２２】第９のシミュレーションのスループットの時間変化を示すグラフである。
【図２３】第１０のシミュレーションのスループットの時間変化を示すグラフである。
【図２４】第１１のシミュレーションのスループットの時間変化を示すグラフである。
【図２５】第１２のシミュレーションのスループットの時間変化を示すグラフである。
【図２６】第１３のシミュレーションのスループットの時間変化を示すグラフである。
【図２７】第１４のシミュレーションのスループットの時間変化を示すグラフである。
【図２８】第１５のシミュレーションのスループットの時間変化を示すグラフである。
【図２９】第７、第８、および第１５のシミュレーションにおける転送量を示す図である。
【図３０】第９乃至第１１のシミュレーションにおける転送量を示す図である。
【図３１】第１２乃至第１４のシミュレーションにおける転送量を示す図である。
【図３２】第７乃至第１５のシミュレーションにおける転送量を示すグラフである。
【符号の説明】
【０１９９】
２０　通信装置，　６３　輻輳ウィンドウ決定部，　６４　送信部

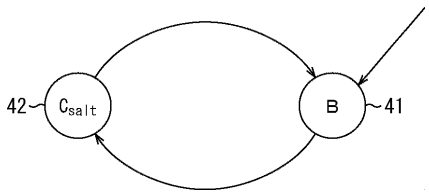
【図１】
図1



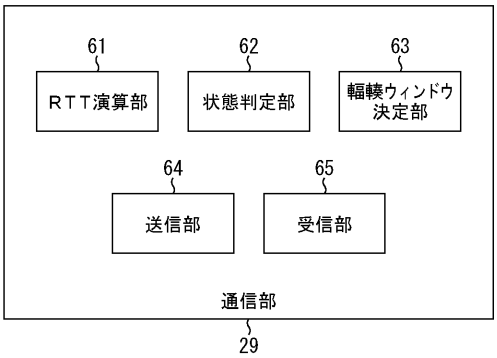
【図２】
図2



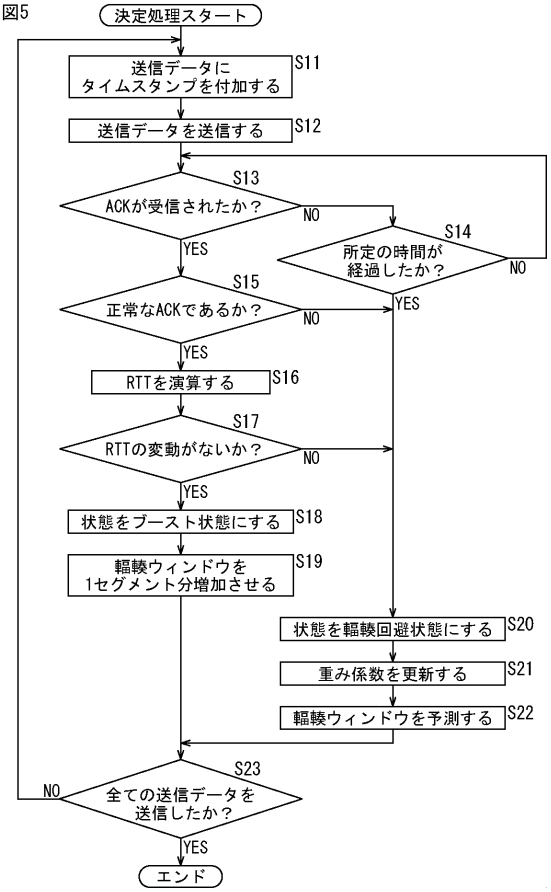
【図３】
図3



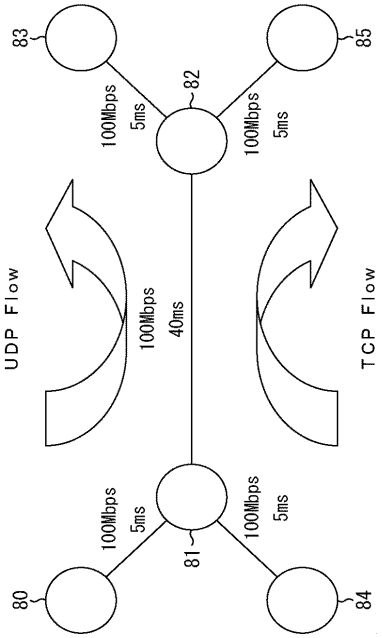
【図４】
図4



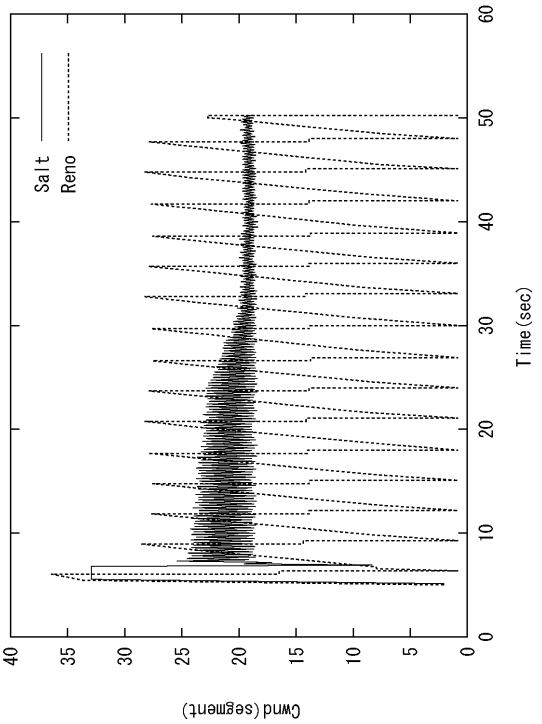
【図５】
図5



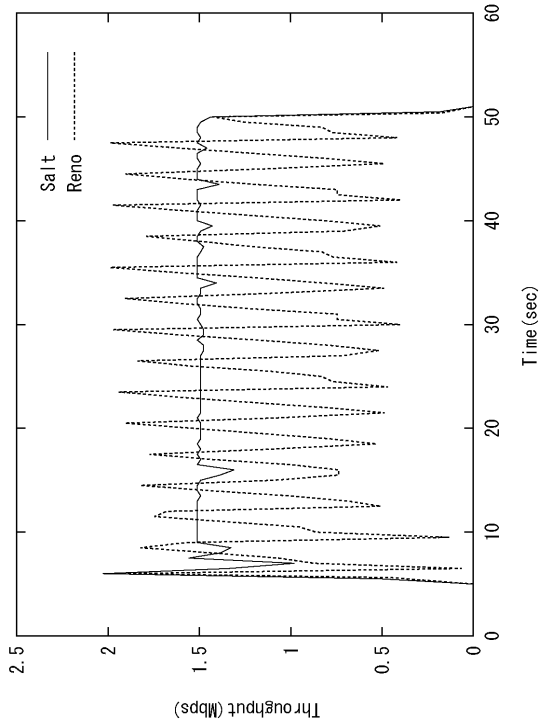
【図６】
図6



【図７】
図7



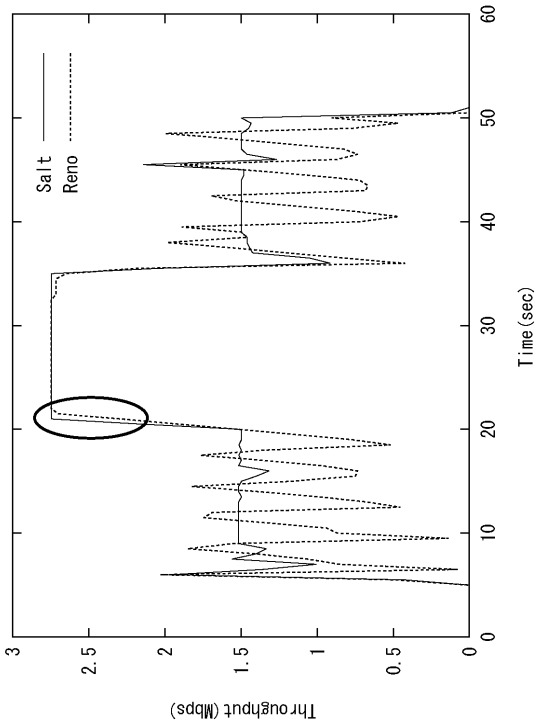
【図８】
図8



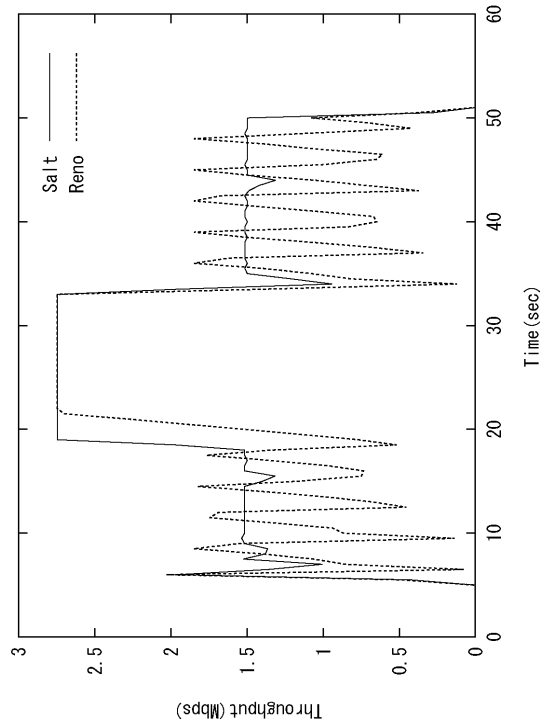
【図９】
図9

	平均スループット(Mbps)	最大スループット(Mbps)	転送データ量(MB)
Reno	1.084	2.030	6.096
Salt	1.428	2.030	8.030
改善率	31.7%	0.0%	31.7%

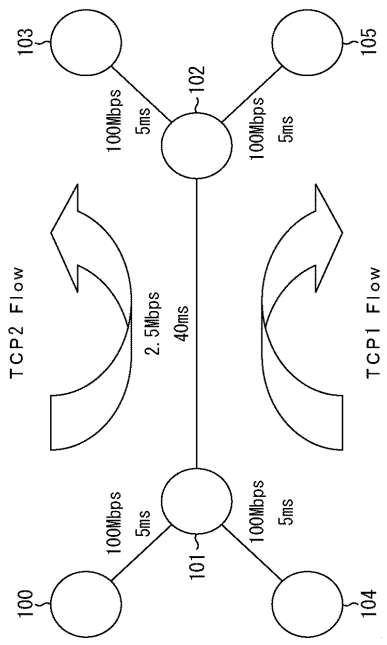
【図１０】
図10



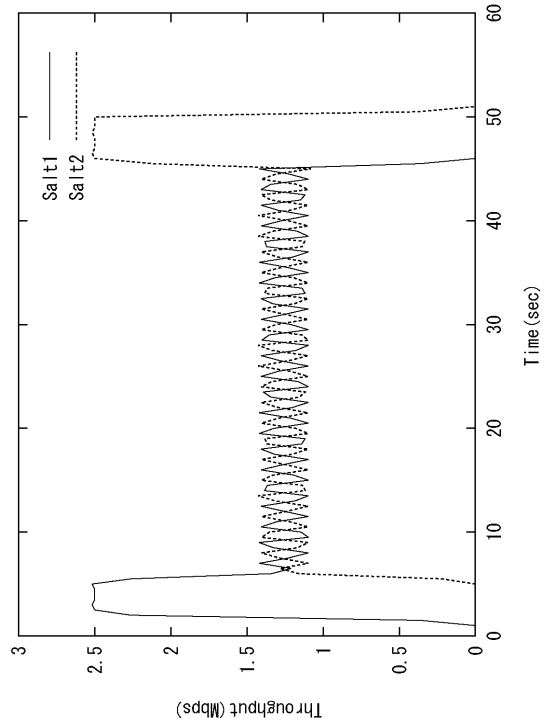
【図１１】
図11



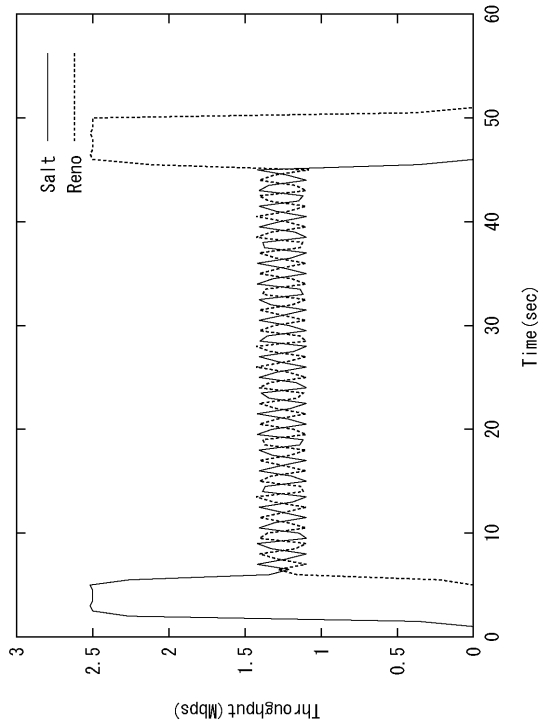
【図１２】
図12



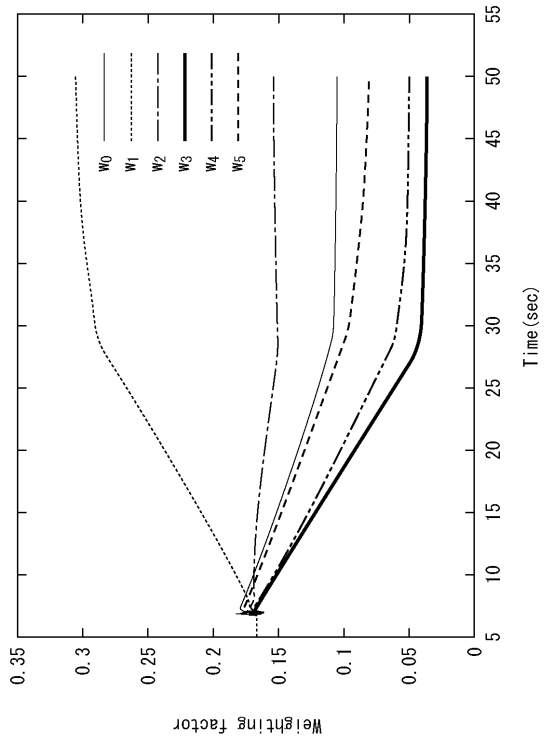
【図１３】
図13



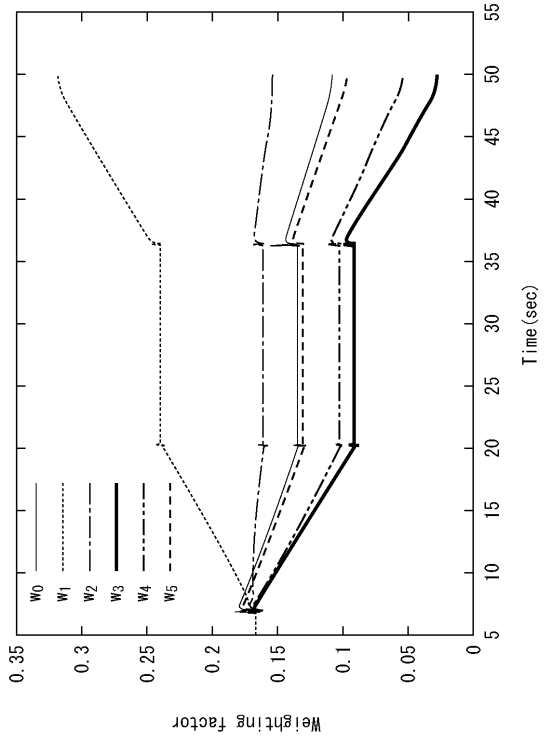
【図１４】
図14



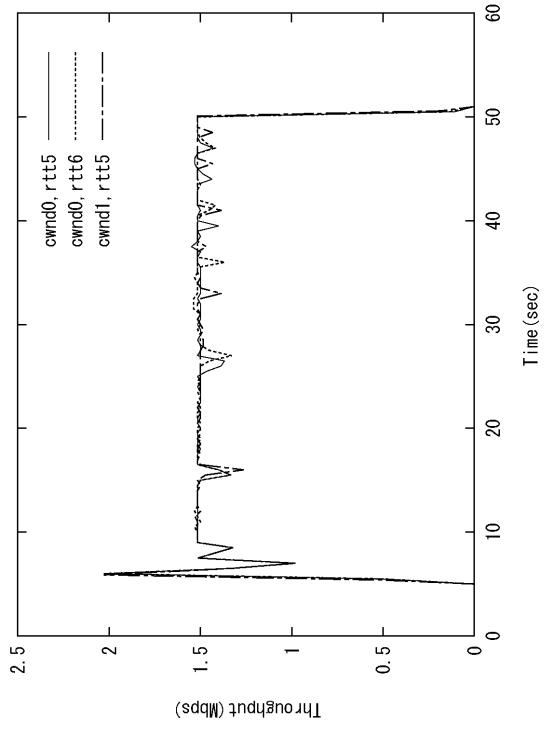
【図１５】
図15



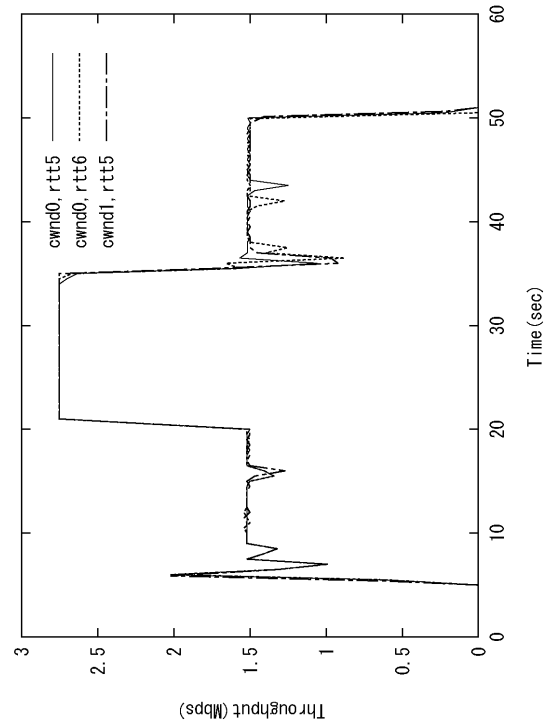
【図１６】
図16



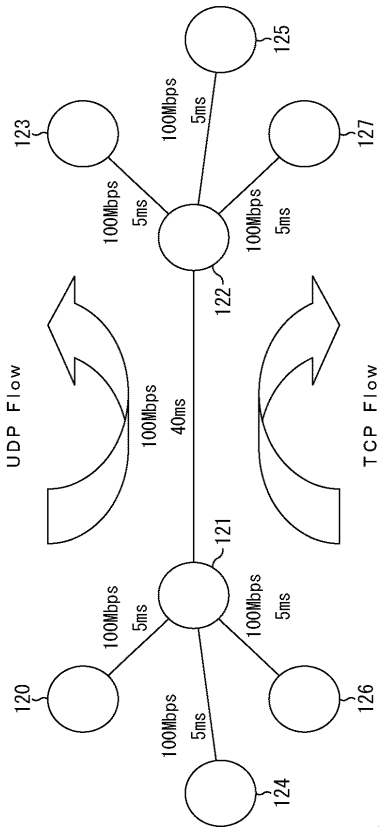
【図１７】
図17



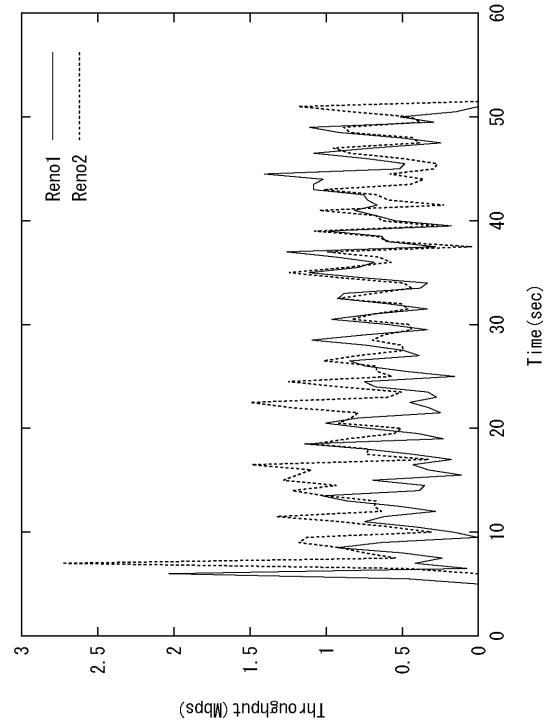
【図１８】
図18



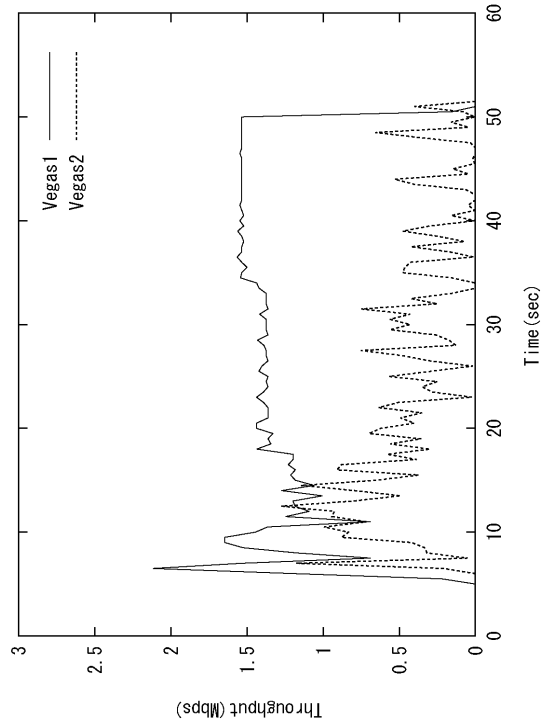
【図１９】
図19



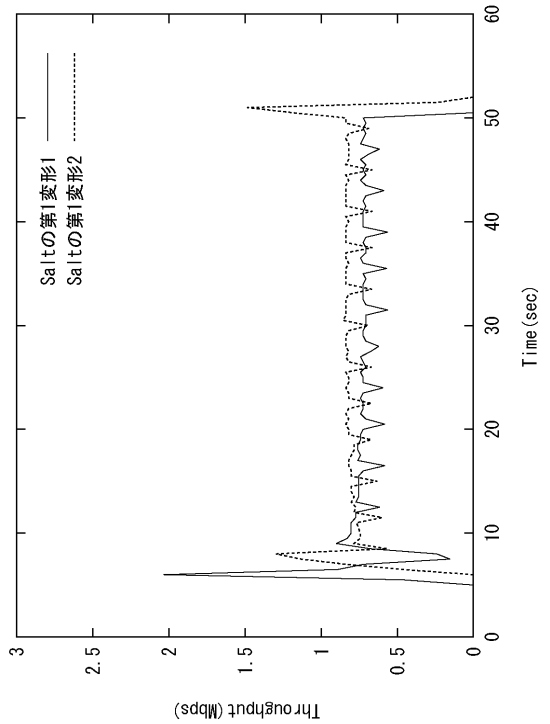
【図２０】
図20



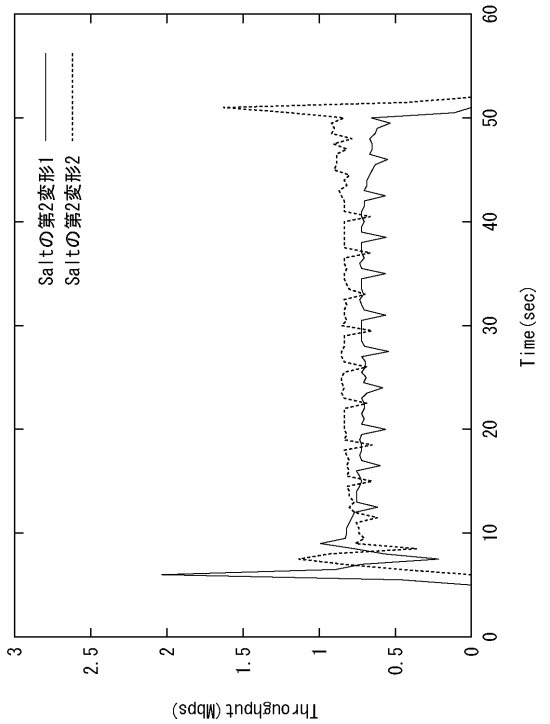
【図２１】
図21



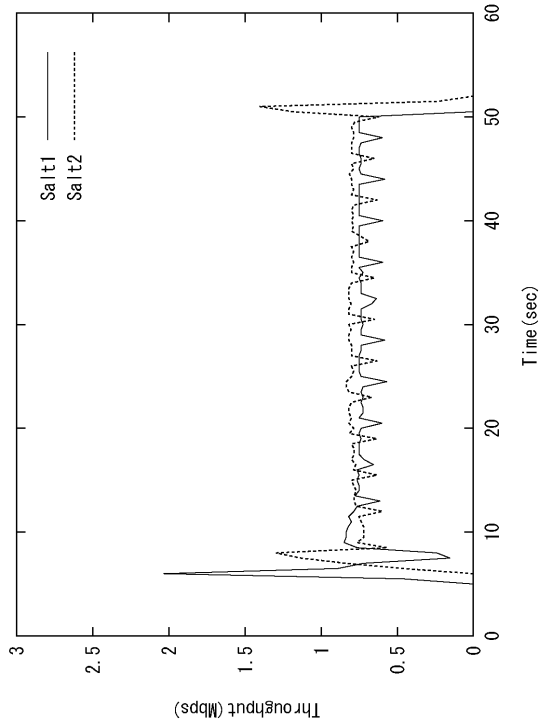
【図２２】
図22



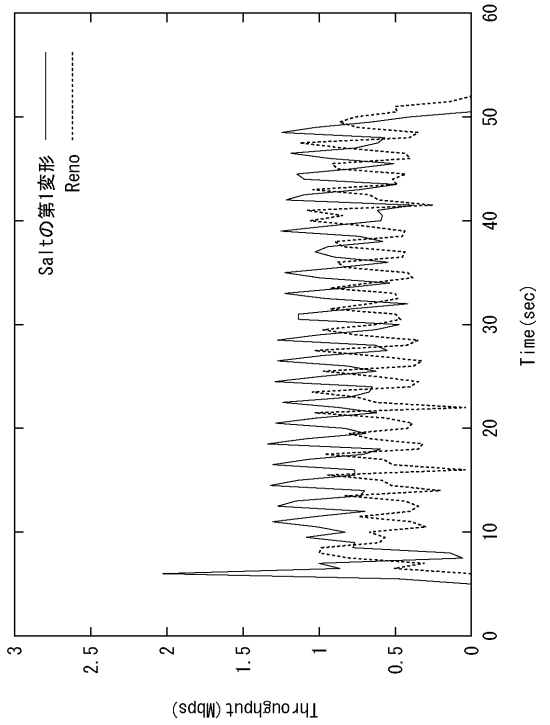
【図２３】
図23



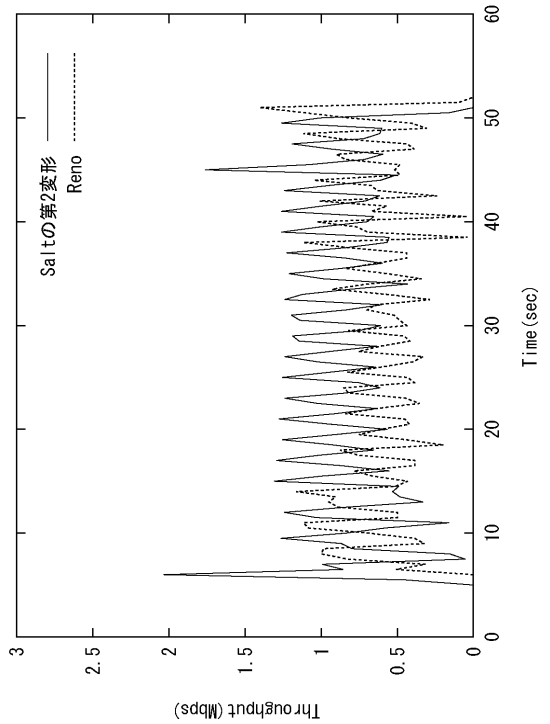
【図２４】
図24



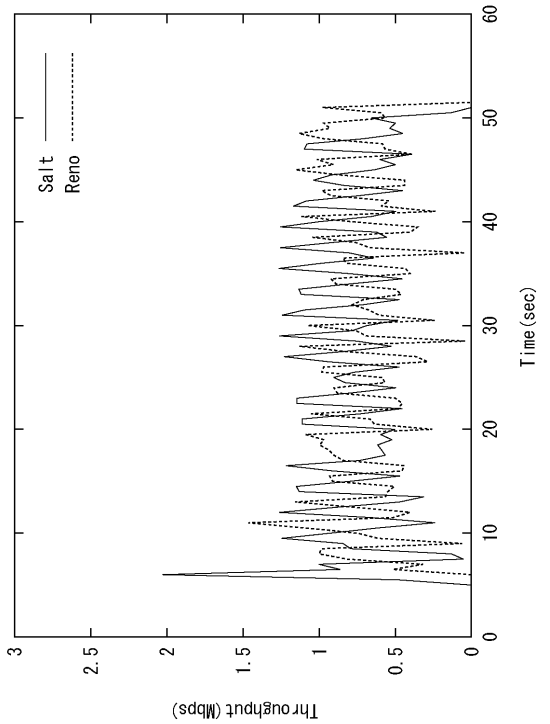
【図２５】
図25



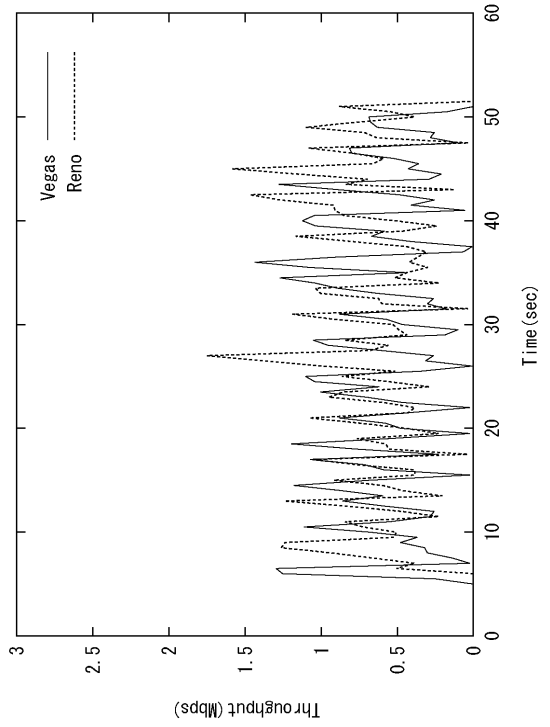
【図２６】
図26



【図２７】
図27



【図２８】
図28



【図２９】
図29

	TCP1	TCP2	TCP1+TCP2	|TCP1−TCP2|
第7のシミュレーション (Reno vs. Reno)	3410	4104	7514	694
第8のシミュレーション (Vegas vs. Vegas)	7851	2176	10027	5675
第15のシミュレーション (Vegas vs. Reno)	3320	3700	7020	380

※全てbyte単位

【図３０】

図30

	TCP1	TCP2	TCP1+TCP2	|TCP1−TCP2|
第9のシミュレーション (Saltの第1変形 vs. Saltの第1変形)	3849	4397	8246	548
第10のシミュレーション (Saltの第2変形 vs. Saltの第2変形)	3818	4462	8280	644
第11のシミュレーション (Salt vs. Salt)	3960	4273	8233	313

※全てbyte単位

【図３１】

図31

	TCP1	TCP2	TCP1+TCP2	|TCP1−TCP2|
第12のシミュレーション (Saltの第1変形 vs. Reno)	4743	3352	8095	1391
第13のシミュレーション (Saltの第2変形 vs. Reno)	4783	3456	8239	1327
第14のシミュレーション (Salt vs. Reno)	4316	3824	8140	492

※全てbyte単位

【図３２】

図32