計測自動制御学会東北支部 第 217 回研究研集会(2004.7.15)
資料番号 217-3

Linux による Serial-Ethernet 変換器を用いた遠隔制御システムの構築

The Construction of Remote Control System Using Serial-Ethernet Converter by Linux

○ 橋元直樹*，秋山宜万*，松尾健史*，三浦武*，谷口敏幸*
○ Naoki Hashimoto*，Yoshikazu Akiyama*，Kenshi Matsuo*，Takeshi Miura *，Toshiyuki Taniguchi*

*秋田大学
*Akita University

キーワード：遠隔制御(remote control)，通信遅延(time delay)，パケット損失(packet loss)，
リナックス(linux)，シリアルーイーサネット変換器(serial-ethernet converter)

連絡先：〒010-8502 秋田県秋田市手形学園町 1-1 秋田大学工学資源学部 電気電子工学科

三浦　武，TEL：(018)889-2329，FAX：(018)837-0406，E-mail：miura@ipc.akita-u.ac.jp

1．はじめに

近年の情報通信技術およびインターネットをはじめとするその利用技術には目覚しいものがあるが, その展開の一つとして情報通信ネットワークと制御理論の融合が注目されてきている [1)].

従来の制御では, 制御対象と制御装置が何十メートルも離れて置かれている場合でも制御信号の伝達, すなわち通信の問題は制御とは切り離して考えるのが普通であった. 制御の側では, 通信は遅れなしに理想的に行われる. すなわち通信路容量は無限大であるとして設計し, 一方, 通信路はなるべく遅れを少なくするように設計する. このように, 両方の問題を分けて考えていた [2)].

近年の技術進歩に伴って, プラントなどの新設や保守を効率的に行うため複数の制御装置を共通の線で結び, 標準化された信号形式と伝送手順を用いて, その間の通信を行い, 分散した制御システムを統一する分散制御システムへの要求が高まってきている. しかし, ネットワークを制御のための信号を伝える媒体として使おうとすると, 制御装置と制御対象はパケット通信による計測･制御信号の送受信を行うため, ネットワークの込み具合による信号の遅れが無視できないだけではなく, どれだけ遅れるかを予測できないなどの「通信遅延」や, トラフィックなどの通信環境によって「パケット損失」などの問題が生じ, 制御システムに大きな影響を与えると考えられる. このようなことから, 制御システムにおいて通信の問題を切り離して考えることができなくなった.

本研究では実際に遠隔制御システムを構築し, 遠隔制御の基礎実験を行った. また, 計測・制御データの転送においてパケット損失が発

生した場合, 制御システムに与える影響について考察した.

なお, 本研究では低コスト性および汎用性を考慮して, 制御システムの OS にはフリーかつオープンソースである Linux とし，近年高速, 低価格化が進んでいる Ethernet, また現在のネットワークにおいて最も普及し標準的なプロトコルである TCP/IP を用いている．また, FA 業界の標準であるシリアル機器を統一，管理を行う分散制御を想定し Serial-Ethernet 変換器を用いて遠隔制御システムを構築した.

2．Linux によるリアルタイム実行

制御において, 信号のサンプリング間隔の精度および揺れ幅すなわち安定性はデータの解析結果に大きな影響を与えるため, センサ信号を読み取り, 制御演算を行い, アクチュエータに制御信号を送るなどの処理をリアルタイムで行うことが非常に重要である. ここで言うリアルタイムとは処理の実行開始が指示されてから処理終了までの時間が規定され, 実時間制約を保障するという意味である. 近年の制御手法の多くはこのように一定周期で制御則を実行することが一般的である．OS のないシステムや MS-DOS のようなシングルタスクの OS では，CPU を自由に使用可能なため，一定周期で制御則を実行する点には問題がない. それに対して, Windows や Linux のようなマルチタスクの OS では CPU を使える時間が原則として不定であるため, 一定周期で処理を実行できない可能性がある．しかし，本研究では低コストにシステムを構築するため, フリーでオープンソースなOSであるLinuxを用いることを考える．はじめに，この Linux を制御用の OS として使用することが可能であるかを検討する.

2-1．RDTSC の精度

CPU に内蔵されている 64bit カウンタ（Time Stamp Counter）より周波数クロックを取得する RDTSC (Read Time Stamp Counter) 命令を利用して一定周期で処理を実行させる．RDTSC は Linux の時間管理に用いられており，分解能は 1CPU クロック，安定性はコンピュータの持つ水晶発振器に依存する．本研究では Intel 社の Celeron プロセッサ 1.0[GHz]を搭載した PC/AT 互換パーソナルコンピュータを使用しており，分解能は理論上 1[ns]となる．しかし，クロックの精度は利用環境にも著しく影響されるので理論通り 1[ns]とはならない．よってここで, RDTSC の精度を確認するため RDTSC で 1[μs](1000 クロック)の空ループを 1[s](1000000 クロック)間実行させる．この方法により 1[μs]に生じる極めて小さな誤差も 1000000 回ループさせることによって誤差が蓄積され 1[s]後には測定可能な大きな誤差となって現れる．結果, 1.142074[s]となり，これより RDTSC では 1[μs]で約 150[ns]程度の誤差を生じることが分かった.

2-2． リアルタイム処理アルゴリズム

リアルタイムで処理を行わせるアルゴリズムは，RDTSC によりクロック取得を繰り返しながら, ある目標時間になると処理を実行させるものである. 最初に基準となるクロックを取得し, 変数に取り込む. その変数に周期時間を加え目標時間とし，その後 RDTSC で時間取得を行わせながら処理実行を待機させ, 取得時間が目標時間になると処理を実行する. 以上を繰り返し行わせることによりリアルタイムで処理を行わせる．なお, 取得されたクロックは時間に換算している．Fig.1 にリアルタイムで処理を実行するアルゴリズムのフローチャートを示す.

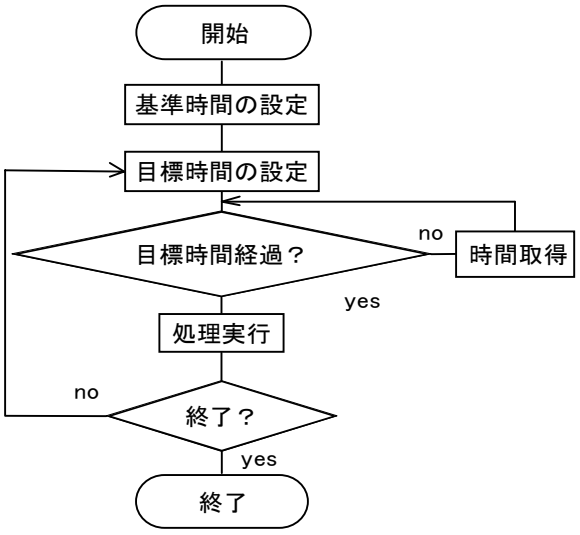


Fig.1　Flowchart

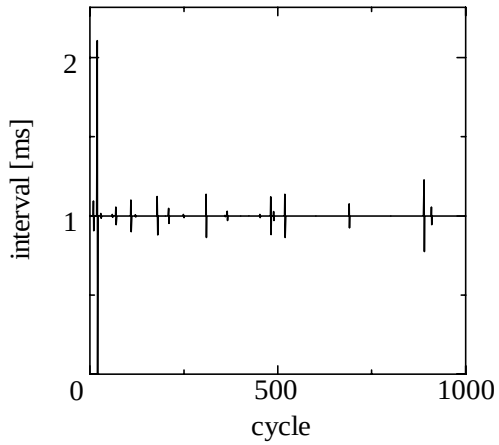


Fig.2　Period of process execution
(1.0[ms]，1000cycles)

2-3.　性能評価実験

上記のアルゴリズムを用いて周期精度および周期時間の揺れ幅を確認する実験を行った．1[ms]周期を目標として，1000 周期の計測を行った．実験結果を Fig.2 に示す．稀に 2[ms]のように周期が長くなる, すなわち実行が遅れるという特徴が見られた．しかし, 目標周期に対する平均誤差は 4[μs]となり, 周期の揺れ幅は平均 1%(誤差±10[μs])以内に収めることができ，また誤差が 1%以上の数は 1000 回中約 20 個となり, 全般に目標周期は精度良く保たれている．

以上の実験結果から周期の揺れ幅も非常に小さく, また周期が長くなる確率も小さいことからリアルタイム機能を持たないマルチタスクな Linux を制御用の OS として使用することが可能であると考えられる．

本手法は, Linux 本体にはなるべく手を加えず，開発の容易さを重視した手法である．

3．Serial－Ethernet 変換器（XPort）

FA 業界でスタンダードとして使用されているシリアル機器では, まだシリアル通信が主流である．Ethernet コントロールをもたず外部との通信手段としてシリアルを標準装備した CPU は，安価で扱いやすい．また歴史が古いため，蓄積された多くのリソースが存在し，簡単にそれらを放棄するわけにはいかない現状もある [3)]．

XPort はシリアル機器を Ethernet に接続するために必要なハードウェア／ソフトウェアを搭載しており, シリアル機器を Ethernet 経由で完全にコントロール可能にすることができる．その結果，複数のシリアル機器を 1 台の PC で集中管理することや, 1 台のシリアル機器を複数の PC で共有することも可能となる．

本研究では, この XPort を使用し遠隔制御システムを構築する．また, 実際のフィールドでは XPort のシリアルポートに接続されるのは計測器，アクチュエータ，コントローラなどであるが, 本研究では制御の方法論に自由度を持たせるため汎用のコンピュータを用いて制御システムを構築する．

4．通信機構

Linux にはネットワークで接続された他のコンピュータのプロセス間で双方向通信が可能となる socket システムコールが用意されている. ソケットはネットワークで接続された他のコンピュータのプロセス間で通信し, データの読み書きすることによって, 双方向通信が可能である. ソケットはメモリ内に設けられたバッファ領域をプロセス間の通信経路にしている.

通信手順は，まず socket( )でプロセス間通信のための端点を作成する．そして connect( )で Serial－Ethernet 変換器に対して接続要求を行う. 接続に成功すると通信用ソケットを読み書きすることによってデータの送受信を行う. また本研究では, 現在最も普及しているネットワークプロトコル TCP/IP を用いる.

Linux でシリアル通信を行う場合，デバイスファイルを利用する. 本研究においてシリアル側では，シリアルポートのデバイスファイル /dev/ttyS0 を読み書きすることによってデータの送受信を行う.

5．遅延特性の計測

本研究で使用する Serial－Ethernet 変換器の性能およびネットワークのパケット遅延特性を評価するため以下の実験を行った.

5-1． 性能評価実験

評価するネットワークに接続されたコンピュータ間に連続的にパケットを流す ping－pong 実験により, パケットの往復時間（RTT：Round Trip Time）すなわち通信遅延を計測し, Serial－Ethernet 変換器の性能およびネットワークの特徴付けを行う. 本研究では，クロスケーブルを用いて XPort を介し 2 台の Celeron 1.0[GHz]の PC を接続し RTT の計測を行う. クライアントはサーバに向けて 4Byte の数値データを送信し, サーバはデータを受け取り, そのままクライアントに向けてエコーバックすることによって RTT の計測を行った. また, 送信データのヘッダにはパケット損失を確認

Table.1 Statistic result [ms]

average	30．073
maximum	54．320
minimum	26．023
standard deviation	290938．4
medium	30．068

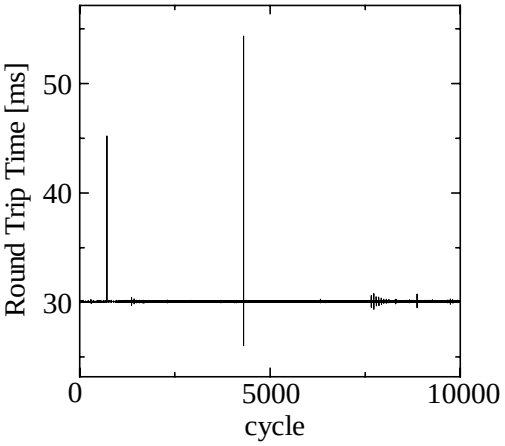


Fig.3　Round Trip Time

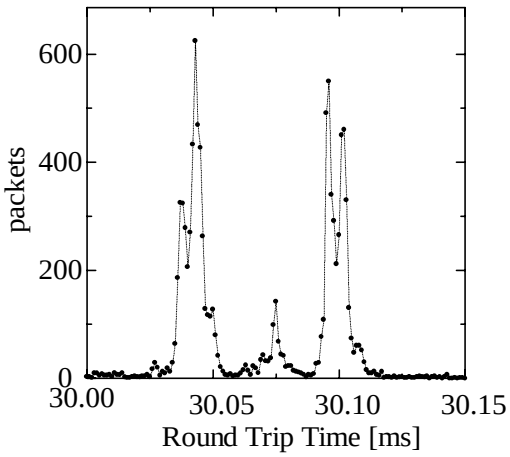


Fig.4　Histogram

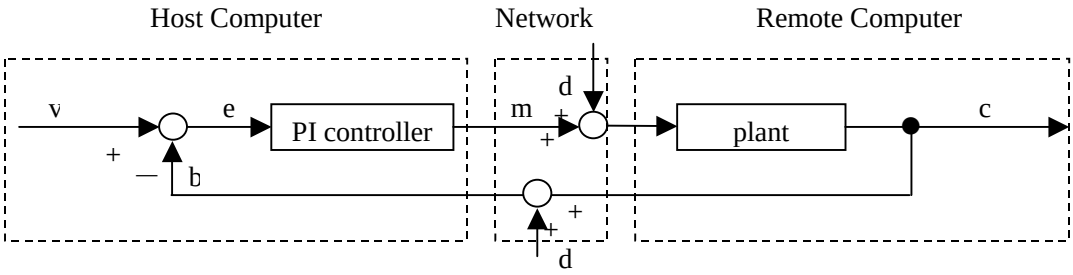


Fig.5　Remote control model

するために送信番号を組み込んだ. なお, 試行回数は 10000 回，PC 間の距離は約 1.5[m]である.

5-2.　実験結果および考察

統計値を Table.1 に, RTT の計測結果を Fig.3 にそれぞれ示す. これらより, ほぼ全てのパケットが遅延 30[ms]付近に集中しており，RTT の平均値は約 30[ms]であることが分かる. また，ごく稀に 50[ms]程度の遅延が生じていることが確認できる. Fig.4 は横軸を遅延時間, 縦軸をパケット数としたヒストグラムである. ただし，遅延時間は 30[ms]付近に限定してある. この図から分かるようにパケットの遅延特性は正規分布や指数分布ではなく 3 つ山型になっている. また本研究においてパケット損失は起こらなかった. なお, 他のトラフィックが存在しないため評価結果には再現性があることが予想されるが，実際に同じ測定を 10 回行い再現性があったことを確認した.

6. DC サーボモータの PI 遠隔制御

リモートコンピュータとホストコンピュータとの間で遠隔フィードバック制御を行った. なお，RTT の平均値が約 30[ms]であることを考慮して制御周期は 50[ms]としてある.

制御対象には位置, 速度制御に優れる DC サーボモータとし, 制御方式は現在実用されている方式の 80%以上を占める PI 制御を用いた. なお本研究では, ハブやルーターなどの通信機器を介さずに 2 台のコンピュータを Serial－Ethernet 変換器経由で直結しデータのやりとりをする閉ざされたネットワークで実験を行う. Fig.5 に本研究で構築した遠隔制御モデルを示す.

6-1. 実験システム

本研究での実験システムを Fig.6 に示す. リモートコンピュータがドライバとなりモータを駆動し, ホストコンピュータが制御演算を行う役割を担っている.

リモートコンピュータから DA 変換器によってアナログ電圧に変換され, アンプで電力増幅を行い DC サーボモータへ印加し駆動させる. モータを駆動した際の回転子速度はタコジェネレータによって検出し, ローパスフィルタおよび AD 変換器を介してリモートコンピュータに取り込まれる. そして, 速度情報はシリアルポートから送信し XPort で Serial-Ethernet 変換されホストコンピュータへデータ転送される. 速度情報を受信したホストコンピュータでは PI 速度制御演算を行い，モータの印加電圧となる操作量を算出する. そしてその情報を再び XPort 経由でリモートコンピュータへ転送し，モータに電圧を印加する.

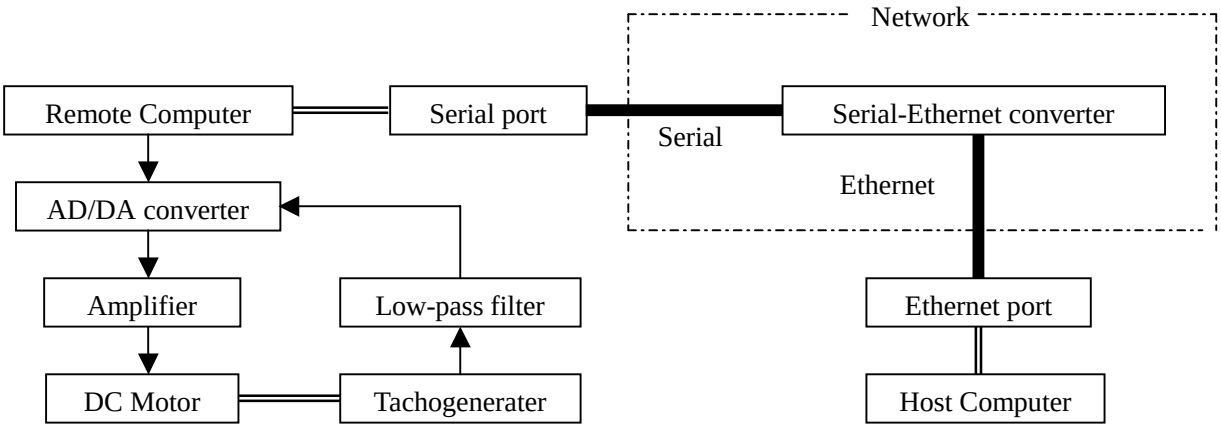


Fig.6　Experiment system

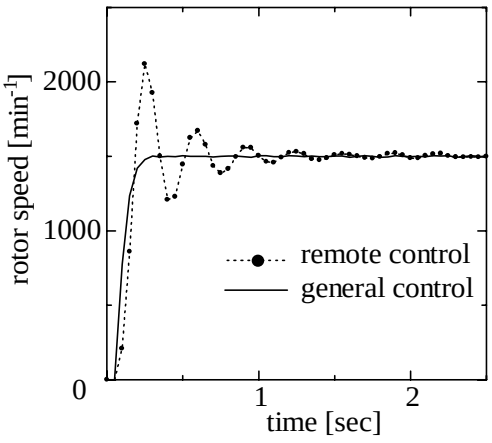


Fig.7　Step response

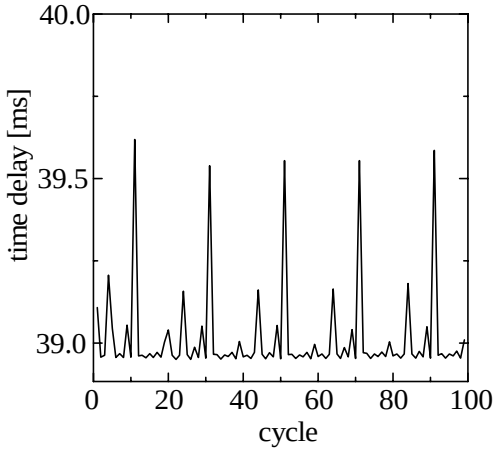


Fig.8　Time delay

6-2.　遠隔制御によるシステムへの影響

本モータに対して想定した設定目標は速度目標値に対してオーバーシュートが生じず, 定小さくなるような PI 制御パラメータを試行錯誤的に設定した. このように制御設計したモータに対して遠隔 PI 速度制御を行った．Fig.7 に速度応答，Fig.8 に計測・制御データの転送時間すなわち通信遅延の結果を示す．Fig.7 より，一般の制御ではオーバーシュートを生じずに目標速度に収束するが, 遠隔制御を行った場合, 目標速度に対して約 30%程度のオーバーシュートを生じ, 速度波形が振動的になっており整定時間が大きくなってしまっていることが分かる．次に Fig.8 より遠隔制御における時間遅れは平均約 39[ms]となり，また，39.5[ms]程度の遅れが周期的に発生していることが分かった．

6-3.　慣性負荷に対する速度応答

慣性負荷 5.0×10^{-5} [N･m･s^2]を取り付けた際の速度応答を Fig.9 に示す．慣性負荷を取り付けることによって, 一般の制御に比べ大きなオーバーシュートが発生し, 整定時間も大きくなり, 制御性能が劣化してしまっていることが確認できる.

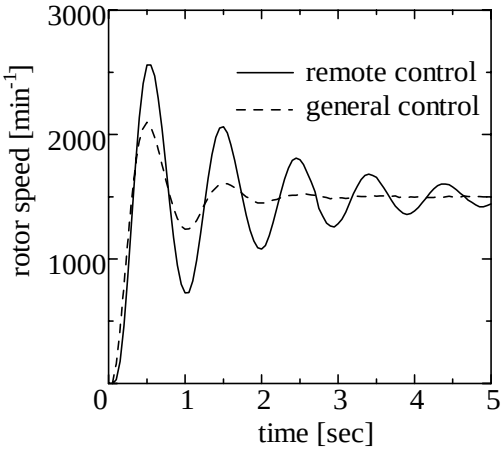


Fig.9 Speed response in inertial load wearing

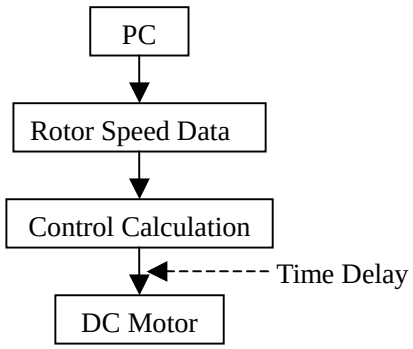


Fig.10 Simulation

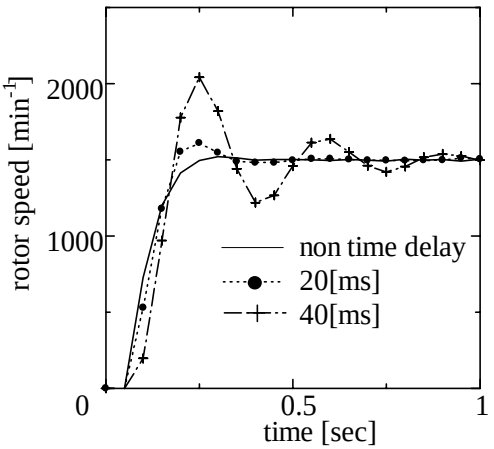


Fig.11 Speed response with respect to various delay time

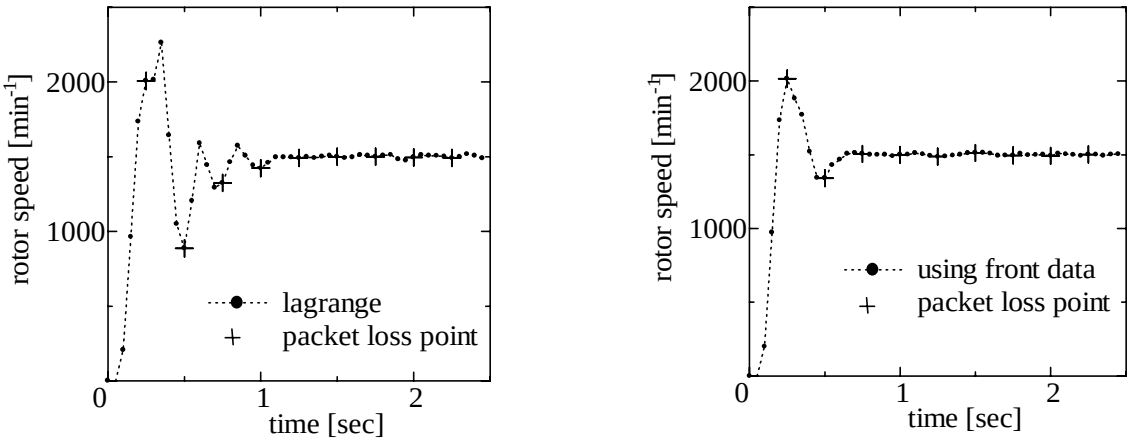
6-4. 通信遅延に対する速度応答

計測･制御データの転送時間すなわち通信遅延の長さによってどのように速度応答が変化するか確認するため 1 台の PC で模擬的な実験を行った．Fig.10 に示すように，速度情報を取得し PI 制御演算を行った後，モータに印加するまで模擬的に時間遅れを発生させるシステムを構築し, 時間遅れをそれぞれ任意に変化させた場合の速度応答を Fig.11 に示す．なお，遅延時間は 0，20，40[ms]の 3 パターンとし，制御周期は 50[ms]である．結果より，時間遅れを大きくしていくと徐々にオーバーシュートも大きくなっており, 遅延時間の長さが制御特性に影響することが分かった．

6-5. パケット損失によるシステムへの影響

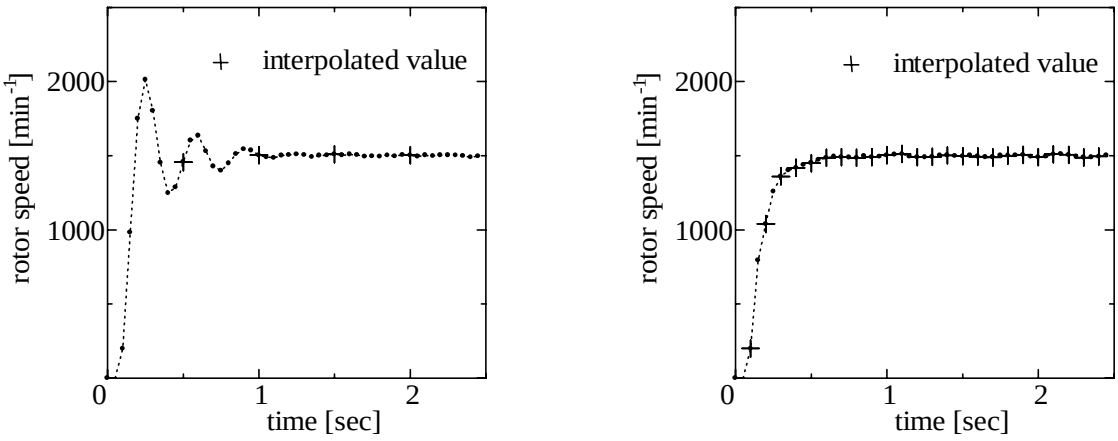
これまでの実験ではデータ転送によるパケット損失は生じなかった. しかし実際にインターネットを利用した遠隔制御を行った場合にはトラフィックなどの通信環境によってデータが損失する可能性が考えられる. このようにパケット損失が発生したとき, どのように速度応答特性が変化するか実験を行った．

パケット損失を一定間隔および過渡状態においてランダムに発生させるシステムをそれぞれ構築し, パケット損失間隔と個数をそれぞれ変えて実験を行う．ここで，損失したデータをどのように補間するかが問題となるが, 本研究ではまず, 3 次のラグランジュ補間法とパケット損失が発生した 1 つ前の操作量を利用する方法で損失データの補間を行い両者の補間法を比較した．なお，パケット損失は 250[ms]おきに一定間隔で計 10 個発生している．結果を Fig.12 に示す．これより，(a)と(b)どちらの補間法でも速度制御できているものの，(a)のラグランジュ補間では速度波形が大きく振動していることが確認できる．これは，パケット損失が発生するまでの操作量が大きく変動し



(a) Lagrange interpolation (b) Front data interpolation

Fig.12 Speed response



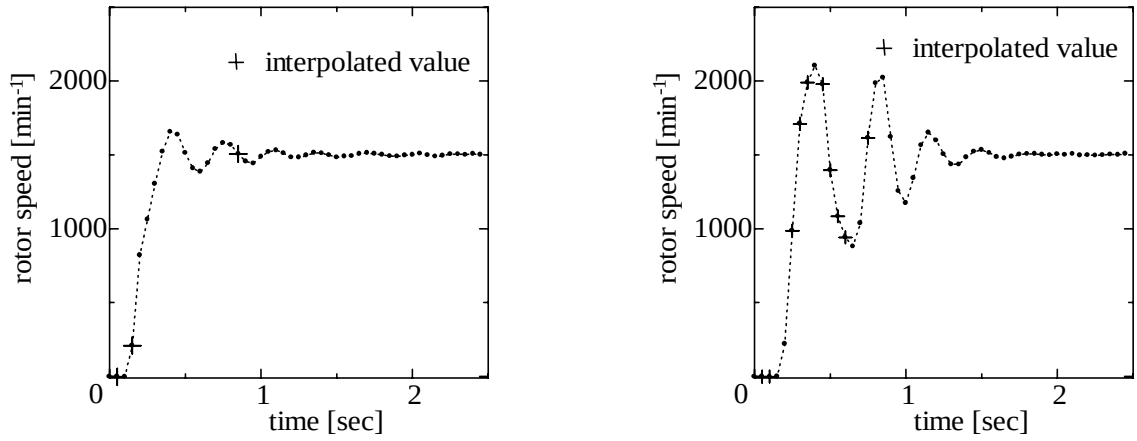
(a) packet loss：4 (b) packet loss：25

Fig.13 Constant packet loss

ていたためラグランジュ法ではうまく補間できなかったと考えられる．よって，以下の実験では(b)のパケット損失が発生した 1 つ前の操作量を利用する方法で損失データの補間をし，遠隔制御を行う．

Fig.13(a)にパケット損失が 500[ms]おきに一定間隔で計 4 個発生した場合，(b)に 100[ms]おきに計 25 個発生した場合の速度応答を示す．(b)では 50%ものデータが損失しているにもかかわらずオーバーシュートも生じず，制御性能を劣化させない結果となった．また(a)ではパケット損失における大きな影響はみられなかった．

定常状態では操作量の変化が少ないため，たとえデータロスが発生しても大きな影響は出ないと考えられる．よって次の実験では過渡状態においてパケット損失が発生した場合にのみ限定して実験を行った．過渡状態を 0~1.0[sec]と定義し，その間でランダムにパケット損失を発生させるシステムを構築し，パケット損失数によって速度応答がどのように変化するか実験を行った．実験結果を Fig.14 に



(a) packet loss：3　　(b) packet loss：10

Fig. 14 Random packet loss

示す．(a)はパケット損失数が 3 個，(b)は 10 個である．結果より，パケット損失数が増加するにつれて整定時間が大きくなり，制御性能が劣化してしまった．これより過渡状態におけるパケット損失が遠隔制御システムにおいて大きな影響を与えることが分かった．

7．おわりに

近年，遠隔制御が非常に大きな注目をされている．しかし，遠隔制御を行う場合，通信遅延やパケット損失が制御システムに大きな影響を与えると考えられる．

本研究では実際に DC サーボモータの遠隔制御を行ったところ，6 章で示したようにオーバーシュートを生じてしまい，整定時間が大きくなる結果となった．このことから通信遅延が制御性能の劣化を引き起こしていることが示された．また，過渡状態において計測・制御データの転送中にパケット損失が発生することによっても，速度波形が大きく振動するなどのようにシステムに対して大きな影響を与えてしまうことも示された．

今後の課題として，制御の性能は通信遅延やパケット損失に依存するため，それらを制御の一部分として組み込まれなければならない．

参考文献

1) 汐月哲夫：情報通信ネットワークと制御理論の融合の可能性，465/468，計測と制御，(2004)
2)木村英紀：制御工学の考え方，206/209，(2002)
3) 川口幸裕：シリアル機器を Ethernet に接続する，139/145，Interface，(2003)