地震研究所彙報
Bull. Earthq. Res. Inst.
Univ. Tokyo
Vol. 84（2009）pp. 89–105

首都圏地震観測網（MeSO-net）のデータ伝送方式について
―自律協調型データ送信手順（ACT protocol）の開発―

森田裕一[1)]*・酒井慎一[1)]・中川茂樹[1)]・笠原敬司[1)]・平田　直[1)]・鏡　弘道[2)]・加藤拓弥[2)]・佐藤峰司[2)]

[1)] 東京大学地震研究所・[2)] 白山工業株式会社

Development of an Intelligent Data Transmission Protocol for MeSO-net System
―Autonomous Cooperative data Transfer (ACT) Protocol―

Yuichi Morita[1)]*, Shin'ichi Sakai[1)], Shigeki Nakagawa[1)], Keiji Kasahara[1)], Naoshi Hirata[1)], Hiromichi Kagami[2)], Takuya Kato[2)] and Minemori Sato[2)]

[1)] Earthquake Research Institute, University of Tokyo, [2)] Hakusan Corporation

Abstract

The Autonomous Cooperative data Transfer Protocol (ACT protocol) is one of the data transfer protocols based on UDP/IP developed for the Metropolitan Seismic Observation network (MeSO-net). It has a robust re-send function to prevent data from being lost in the Internet route. Moreover, it also has a function to adjust the data transmission rate considering the quality of the Internet route and the load on the data-receiving system. If the Internet route is too busy to send all data, the transmission rate from the observation station is decreased automatically to ease the Internet traffic jam, and data that cannot be sent because of the limitation on the transmission rate are stored at each station. After conditions recover, the stored data are sent automatically at an increasing transmission rate and gradually catch up with real-time data. The transmission rate is decided by data transmission equipment at the station using pre-loaded algorithms, not at the data center. Therefore, the load at data center equipment is not so high even if the equipment receives data from hundreds of stations. We describe an overview of the ACT protocol, flow charts, and data format used in the protocol. We also demonstrate that the ACT protocol is very powerful for the vast size of the seismograph network composed of several hundred stations. The ACT protocol is generally applicable and can be used for data transmission systems in other scientific fields.

Key words: Seismometry, Seismic network, Data transmission, Internet Protocol, MeSO-net

1. はじめに

多数の観測点からの地震データが，正確な時刻情報と共にほぼ実時間で研究所等に設置されたデータセンターに収集されることは，今や当たり前のこととなっている．このような便利な状況になったのは，約 20 年前に実用化された 2 つの技術が背景となっている．ひとつは GPS 受信機を利用した安価で高精度な時計（森田・他，1993）の導入により，個々の観測点で誤差が 1μ 秒を切る高精度の時刻管理が可能となったことである．これにより，観測点側でデータに正確なタイムスタンプが付けられ，データ伝送の遅延時間にばらつきがあっても研究を遂行する上で問題にならなくなった．もうひとつは，当時黎明期にあったインターネット技術を採用したデータ伝送システム“WIN システム”（卜部，1994）が開発されたことである．初動到達時刻が重要な情報である地震データでは，その当時は観測点間の時刻同期が難しかっ

*e-mail: morita@eri.u-tokyo.ac.jp（1-1-1 Yayoi, Bunkyo-ku, Tokyo 113-0032, Japan）

たので，利用料が高価ではあるが伝送遅延が一定である専用回線を使用してデータ伝送していた．しかし，上記の 2 つの技術開発で伝送遅延の問題は解決し，安価なインターネットを利用した地震データの伝送が利用できるようになった．処理装置との相性の良いインターネットを利用したデータ伝送は，多数の観測点からの多量のデータを容易に処理することを可能にし，現在日本国内で普及しているデータ伝送システムの原型がほぼ完成した．1990 年代の日本の地震観測データの伝送技術は，インターネット技術を積極的に取り込み，世界の最先端をリードするものであったと言える（森田・大湊，2005）．

この 20 年間のインターネットの普及は著しく，たとえ発展途上国の僻地であっても，衛星回線を利用したインターネットの利用が可能となった．そして，それを用いた地震波形データの伝送は当たり前になってきた（例えば，石原・他，2004）．一方，このようなインターネットの世界的な普及は，異なる社会基盤や社会情勢に最適なように導入されたため，多様なインターネット環境が出現するようになった．使用料が高価ではあるが，単位時間当たりのデータ伝送量（帯域）を保証するだけでなく，データの伝送遺失がほとんどない高品質の回線から，安価ではあるが帯域保証がなく，伝送遺失が多い低品質の回線まで，世界中には色々なインターネット回線が混在している．帯域保証のない回線では，公表されている伝送容量は最高値であり，実効値は時間帯によっては最高値の 10%程度にまで落ちることが頻繁に起こる．また，衛星回線インターネットでは，伝送遅延は地上回線の数倍から数十倍程度大きい．このような，インターネット環境の変化に対して，主要部分が 20 年以上前に開発された WIN システムは残念ながらうまく対応しておらず，回線状況の悪い観測点からの伝送では，データの欠落が高い頻度で発生する．これは，連続データの一部でも欠落することが研究遂行上大きな障害になる火山性微動や地震波干渉法の研究などでは，大変不都合であると言える．

観測点側の機器にデータを一時的に保存する装置を設置すれば，データの欠落を回復する手段はあるが（Uehira, 2009），首都圏に約 400 点の地震計を設置する計画である首都直下地震防災・減災特別プロジェクト（平田・他，2009）では，限られたプロジェクト期間，少ない人的資源で大量のデータを確実に取得することが不可欠であり，既存のデータ伝送手法を用いて，観測点側に装置を付加してデータ欠落を回復すると言う対応は，観測網運営に多大な労力が必要になると判断した．そのため，多様な品質のインターネット回線を利用して送られてくる多数の観測点からのデータを，単一のデータ伝送手順を用いて，データパケットの欠落がなく，安定かつ確実に人手をかけずにデータを受信できる新たなデータ伝送方式を開発した．このデータ伝送方式は，多様な環境のインターネットに対応できるものである．首都直下地震防災・減災特別プロジェクトでは，ここで紹介するデータ伝送プロトコルの開発以外にも，観測装置，設置方法（笠原・他，2009），データアーカイブ（中川・他，2009）などの新たな技術の開発を行い，保守性の高い高精度の地震観測を実現できる新たな観測システムを構築した．

ここで開発したデータ伝送方式は，自律協調型データ伝送プロトコル（Autonomous Cooperative data Transfer Protocol，以降「ACT プロトコル」とする）と命名され，以下の特長を持つ．1）回線の混雑状況やデータ受信側装置の過負荷によってデータ伝送が滞った時には，観測点に設置したデータ送信装置の判断によって単位時間当たりのデータ送信量（送信レート）を自動的に減らす機能を有する．これにより，回線の混雑の緩和や受信装置の負荷の低減を実現し，ネットワークやシステムが正常な状態に回復するまで待機する．正常状態に回復した際には，自動的に送信レートを増加させて送信装置に蓄えられたデータを送信し，伝送データが実時間に回復するまで可能な限り高速で送信する．2）データ送信装置は，データが正常に行われたか否かの情報をデータ受信装置から受け取り，送信が未完了のデータは完了するまで何度でも再送信する．これにより，データの欠落はデータ送信装置の記録容量を越えるまで原理的に起こらない．

特に，1）の特長は観測点に設置されたデータ送信装置が自律的に振る舞い，観測ネットワーク全体が調和して安定して動作することを追求したもので，観測データの送信方式として新たな考え方を取り入れたものといえる．現実的に，この手順を利用した首都直下地震防災減災特別プロジェクトの首都圏地震観測網では，2009 年 10 月時点で 178 観測点から 200 Hz サンプリング 3 成分の地震波形データを連続して伝送しているが，実時間から遅れて伝送されたデータも含めると全点，全チャンネルで，1 サンプルもデータの欠落なくデータが伝送されている（酒井・平田，2009）．このように，実際の運用が継続すると共に，このデータ伝送手順の高い安定性と信頼性は実証されつつある．

本報告では，この手順の基本的な考え方を記載し，テータ受信プログラムの公開を行う．その目的は，このデータ伝送手順が地震観測や他のデータの伝送手順として広

Table 1. Data format using ACT protocol

項目	開始オフセット	サイズ（バイト）	備考
識別記号	0	4	0x31415926 固定
通番	4	8	0 から半永久的に増加
ACK 返信単位	12	2	ACK 返信単位（表 3 参照）
データ種別	14	2	データの種類を示す（表 2 参照）
データ長	16	2	データ長 (ℓ)
データ	18	ℓ	データ実体
CRC*	18＋ℓ	2	生成多項式は $X^{16}+X^{12}+X^3+X+1$

*CRC 値はデータをチェックするために付加する．送信装置は，先頭から CRC 値の直前まで上記の生成多項式に従い演算し，その数値を書き込んで送信する．受信装置は，受信データに対して同様の演算を行い，送信されてきた CRC 値と一致するかを確認する．一致しない場合は不良データとして棄却される．

く普及するように，また，この手順の考え方を更に発展させて一層有用なデータ伝送手段が生まれることにより，地震観測技術の発展に寄与するためである．

2. ACT プロトコルの伝送方式

ACT プロトコルを詳述する前に，従来のデータ伝送方式である WIN システムとの類似点と相違点を示す．WIN システムでは，基本的には観測データを UDP 方式でインターネットを介して伝送している．送信中のデータパケットの遺失を自動的に回復する手順である TCP 方式を用いずに，パケット遺失に補償のない UDP 方式を用いているのは，処理装置のデータ伝送プログラムによる負荷を軽くするためである．このため，WIN システムは処理能力低いハードウエアでも良好に動作する．伝送中にデータパケットが遺失される可能性がある UDP 方式の欠点を解決するために，WIN システムでは受信装置から未到達のデータ番号を NACK（Not Acknowledge）信号として送信装置に送信する．送信装置は NACK を受け取ると，そのデータ番号のデータを再度送信してデータの欠落を防ぐ．この方式は，未到達のデータが少ない場合には，データ受信装置の負荷が少なく，安定して確実に動作する．しかし，未到達のデータが多くなると，NACK の送信が増えるために負荷が増加するという欠点を持っている．また，NACK 信号そのものが伝送中に失われた場合には，データ送信装置はデータの伝送が完了したと認識し，永久にデータは再送されず，最終的にデータは欠落する．更に，どこか 1 ヶ所の観測点でも故障して誤ったデータを送信し始めると．何度も NACK 信号が発信され，ネットワークは混雑する．混雑が進めば，最悪の場合にはシステム全体が停止するおそれもある．

Table 2. Definition of data used in ACT protocol

項目	値
ACK データ	0x0006
コマンド	0x0003
Win 形式地震波形データ	0x00a0
制御用に予約	0x0000 ～ 0x007f

一方，ACT プロトコルは，WIN システムと同様にインターネットを利用して UDP 方式でデータを送受信するが，正常に伝送されたデータ番号を ACK（Acknowledge）信号として受信装置から送信装置返信する方式を採用している．但し，データを 1 回受信する毎に ACK 信号を返信していると受信装置の負荷が大きくなるので，N 回分のデータ受信状態をまとめて 1 回の ACK 信号で返信する方式を採用している．N は表 1～3 で示す「ACK 返信単位」としてデータ本体と同時に送信されるもので，送信側で定義できるパラメータである．現在は $N=8$ として運用している．送信装置は，一定時間内に ACK 信号が返信されない場合には，全てのデータを再送する．このような仕組みで，確実なデータ伝送を実現している．この方式は WIN システムに比べるとデータ受信装置の負荷は大きい．しかし，近年のハードウエアの性能の向上を考えると，WIN システム開発当時には負荷が大きくて利用できなかった方法も，現在では十分に利用可能となり，ACT プロトコルのような確実な方法が取れるようになったと言える．

更に，ACT プロトコルでは，回線の混雑やデータ受信装置の過負荷による観測ネットワーク・システム全体が

Table 3.　Data format of ACK (Acknowledge) signal

項目	開始オフセット	サイズ（バイト）	備考
識別記号	0	4	0x31415926 固定
通番	4	8	0 から半永久に増加
ACK 返信単位	12	2	ウインドウサイズ
データ種別	14	2	0x0006 に固定（表２）
データ長	16	2	データ長 (現在 12)
ACK の基底*	18	8	
基底からのオフセットのビットマップ**	26	4	
CRC	30	2	$X^{16}+X^{12}+X^{3}+X+1$

*ACK の基底

例えば，ACK 単位 4 の場合，データ通番 2080，2081，2083 が正常に受信でき，2082 は欠落等により受信できていない場合，ACK 基底は必ず ACK 単位の倍数で定義され，この場合は 2080 となる．例えば 2080 番が正常に受信できていない場合であっても 2080 が基底となる．

**基底からのオフセットのビットマップ

実際に受信完了したデータの通番を基底からのオフセットの値を，ビットマップ列で記述する．例えば 2080，2081，2083 番の受信完了を通知する ACK の例では，基底は 2080 でビットマップは２進数表現で 11010000000000000000000000000000 となる．この場合 ACK 単位が４であるので４ビット以降は無視される．ビットマップを４バイトと固定しているため，ACK 返信単位の最大値は 32 に固定されている．

不安定となることを防止するため，データ送信装置から受信装置への送信レートを動的に変化させる仕組みを取り入れた．例えば，帯域保証のない回線で，多くの利用者が同時に多量のデータ送受信を行い，トラフィック量が回線容量の上限を越えた場合を想定する．このような場合には，回線をつなぐルータ等の通信機器の中でパケットの一部は失われ，結果的に送り先に届かずにデータは失われる場合がある．失われたデータを回復するために観測点からデータの再送を頻繁に行えば，回線の混雑が加速され事態は一層悪化する．また，回線に問題がなくても，データ受信装置に各観測点からのデータが一時的に集中して，処理が追いつかない場合にも，データの多くは失われることになる．これを回復するために各観測点からデータを再送すれば，受信装置の負荷が一層大きくなり，状況を一層悪化させる．このような悪循環を生じる事態を回避するには，データ受信装置から送信装置にデータ送信を一時中断することが有効である．受信装置から送信装置全てに送信一時停止命令を送ることも考えられるが，データ受信処理のために負荷が大きくかかっている装置に，別の仕事をさせることは賢明な対処法とは言い難い．ACT プロトコルでは，このような場合に，データセンターからの命令ではなく観測点側装置の判断で自律的にデータ送信レートを減らし，回線や受信装置の負荷を低減させ，ネットワークやシステムの回復を待つ機能（自律協調機能）を持っている．これにより，観測ネットワーク及びシステム全体が安定して動作する仕組みを確立した．具体的な仕組みについては，以下に述べる．

図 1 は，ACT プロトコルの送信装置のフローチャートを示したものである．現在の送信装置では，1 秒毎にデータパケット（データ伝送の単位となる時系列データの集合）が生成され，0 から始まる番号が付されて送信を待つデータの待ち行列に加えられ，メモリ内に保存される．一定時間ごと（Tx: 現在は 1 秒に設定）に，送信待ち行列にあるデータパケット優先順位を計算し，優先度の高い Nx 個のデータを取り出し，表 1 の書式で UDP 方式を用いてインターネットを介してデータ受信装置に送信する．送信したデータパケットは，送信待ち行列から送信済みデータ列に移される．送信と同時にタイマーをセットし，一定時間（Ta）データ受信装置から ACK 信号が返信されるのを待つ．この間も Tx 時間経過ごとに同様の動作を繰り返す．ACK 信号が返信された時は，それから受信装置で受信に成功したデータ番号を読み取り，それを送信装置の送信データ列から消去する．一方，

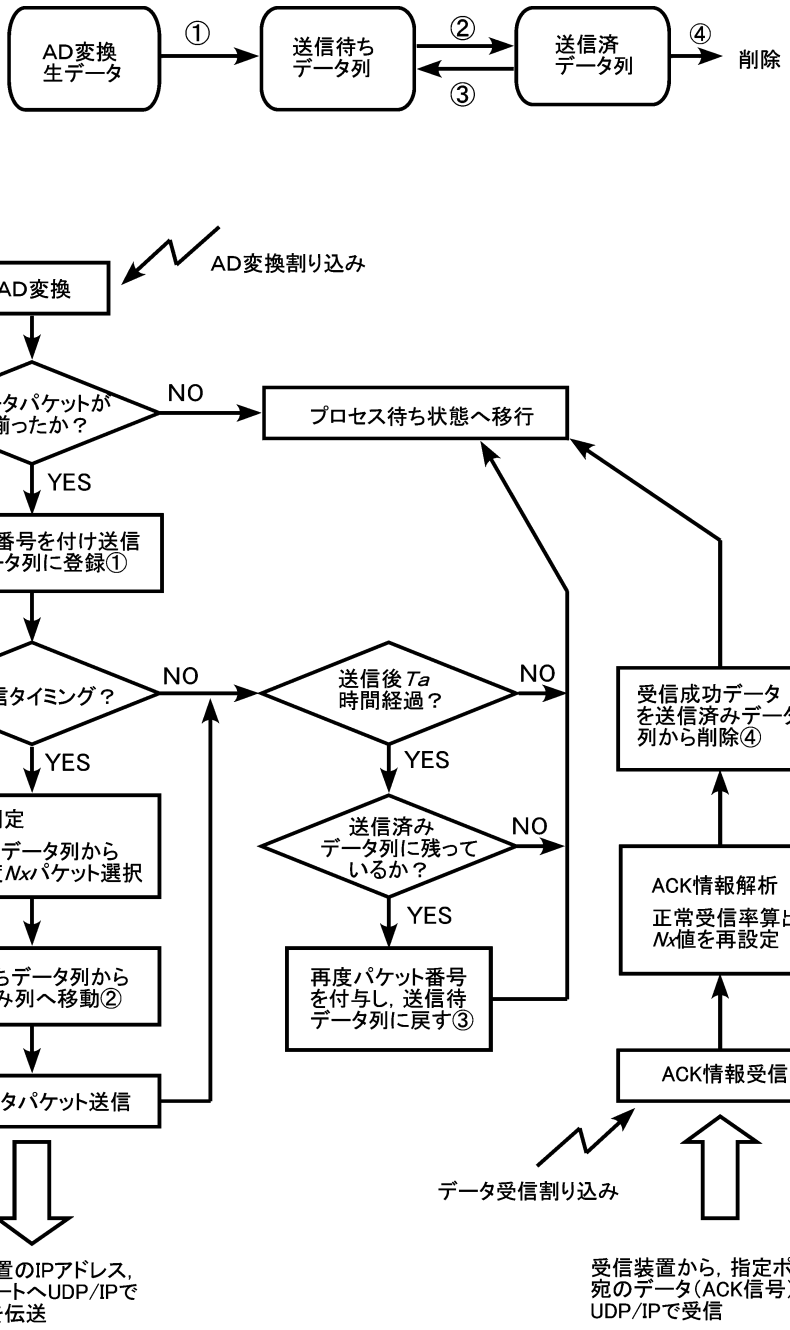

Fig. 1. Flow chart of data-sending system using ACT protocol.

Ta 時間経過しても ACK 信号が返信されないデータは，新たなデータ番号をつけて再度送信待ち行列に加える．データ番号は表 1 のように 8 バイト整数で定義され，1 秒に 100 データパケットが生成される場合でも 6×10^9 年まで数値はあふれが生じず，半永久的にデータ番号を割り振れる．現在は，この数値あふれに対する処理は組み込まれていない．

例えば，何らかの理由で回線が切れた場合には，すべての送信データの ACK 信号が返信されないので，*Ta* 時間経過後にすべて送信待ち行列に戻される．結果として，新たに生成されたデータパケットを加えつつ，未送信のデータパケットは送信待ち行列に残り，メモリ容量一杯まで蓄積されてゆく．回線が復旧した場合には，優先度の高いデータから受信装置に順次送信される．通常，送信レートの上限がデータ生成率を上回るように設定されているので，送信装置に溜まったデータは徐々に減ってゆき，最後にはデータ生成と同時に伝送されるリアルタイム送信になる．データ伝送の優先度の演算は独

立したサブルーチンとして作られている．例えば，最新のものを優先とするとリアルタイム伝送に近づき，古いものを優先にするとデータの生成順にデータが伝送される．データの優先度の演算は，データ送信を止めずに切り替えられるように作られており，大きな地震の直後に回線が不通になった場合，回線復旧後に本震から伝送する，または最新の余震から伝送するなどをデータセンターから制御することができ，研究上最も必要なものを優先して送信することができる．

送信装置は一定時間間隔 *Tx* でデータパケットを *Nx* 個ずつ送信するが，この *Nx* の値を変更することで，データ送信レートを調節している．但し，送信待ち行列にあるデータパケットの数が *Nx* 以下の場合には，それらすべてを送信する．*Nx* は，上限値と下限値を最初に与え，その範囲で以下のような規準で変更する．ACK 信号の情報から受信装置に正常に送信されたデータパケット数と送信装置から送信されたデータパケット数の比（ここでは，「ACK 受信率」と呼ぶ）を一定時間で平均し，これが一定以上であれば上限になるまで *Nx* を徐々に大きくし，一定以下であれば下限になるまで徐々に小さくする．これにより送信レートを制御する．回線の混雑や受信装置の過負荷などの原因により ACK 受信率が下がれば，送信レートを下げて状況の回復を待つという自律協調機能が，このような仕組で実現している．*Nx* をどのように変化させるかについては，伝送遅延時間を基準にする等の他の手法も考えられ，改善の余地はある．しかしながら，現在のところ上記のような ACK 受信率を基準とする最も簡単と思われる方法で，実用上の問題は発生していない．*Nx* の上限値と下限値は回線の容量や，データの生成率，送信装置のメモリ量などから適切な値を予想して与えるが，このパラメータを自動的にチューニングする機能は，現時点では実用化されていない．

図 2 はデータ受信装置のフローチャートである．受信装置では，各観測点から届いたパケットに対して，データ識別番号と循環冗長検査（CRC）値のチェックを最初に行い，不良データは破棄する．その後にデータ識別を行い，それぞれのデータ種別によって独立した処理を行う仕組みとなっている．現在のソフトウエアでは，win フォーマットの地震波形データしか対応していないが，後述のように受信装置のプログラムのソースコードは公開するので，他種類のデータを受信する場合には，その部分を書き加えることで対応が可能である．

地震波形データを受信すると，データ自身をメモリに蓄えると同時に，その観測点，テータ通番を管理するテーブルに情報を書き加える．前回の処理から一定時間が経過するか，テーブルが一杯になると，メモリ上の受信データをハードディスクに書き込み，観測点毎に正常に受信したデータ通番リストを作成し，それに基づき ACK 信号を各観測点に返信する．ACK 信号の書式を表 3 で示す．ACK 信号は，表 1 の書式で各観測点から送信されてきたデータの「ACK 返信単位」に従い，ACK 返信単位毎に返信される．現在のシステムでは，ACK 返信単位を 8 として運用しており，観測点から 8 データパケットが送信されるのに対して，受信装置は ACK 信号が 1 回返信する．現在の書式では，ACK 返信単位は 2 のべき乗でなければならず，その上限を 32 としている．

図 3 に具体的な例を挙げて，データ送受信の仕組みを示す．ここでは表 1 にある ACK 返信単位が 4 に設定されている場合を示す．これは 4 データ毎に ACK 信号を返信することを意味する．例えば，データ通番が 2080 から始まるデータを送信する場合，以下のような流れでデータが送受信される．

1）送信装置が通番 2080，2081，2082，2083，2084 のデータパケットを受信装置に向けて送信する．ここで仮に 2082 番が，伝送路で欠落して，送信装置に到着しなかったとする．
2）受信装置は通番 2080，2081，2083 に対する ACK 信号を返信する．
3）送信装置は ACK を受理した 2080，2081，2083 番のデータを消去する．一方，2082 番のデータは送信済みデータ列に残されている．
4）送信装置は次のデータ送信タイミングが来ると 2085，2086，2087，2088 番のデータを受信装置に送信する．この時，2088 番が欠落すると仮定する．
5）受信側装置 2084，2085，2086，2087 番の ACK 信号を返信する．
6）送信装置は ACK 信号の情報から 2084，2085，2086，2087 番のデータを消去する．この段階で 2082，2088 番が送信済データとして登録されている．
7）一定時間後にタイムアウトしたときは，2082，2088 番のデータは送信失敗と判定され，2082 番には 2089 番，2088 番には 2091 番のデータ通番が再交付され，送信待ちデータ列に加えられる．この例では，2090 番は新たに生成されたデータである．
8）送信装置は 2089 番として 2082 番の再送，2090 番として新規のデータ，2091 番として 2088 番の再送データを転送する．
9）これらのデータが正常に受信できた時には，受信装置は 2089，2090，2091 番に対する ACK 信号を返信する．

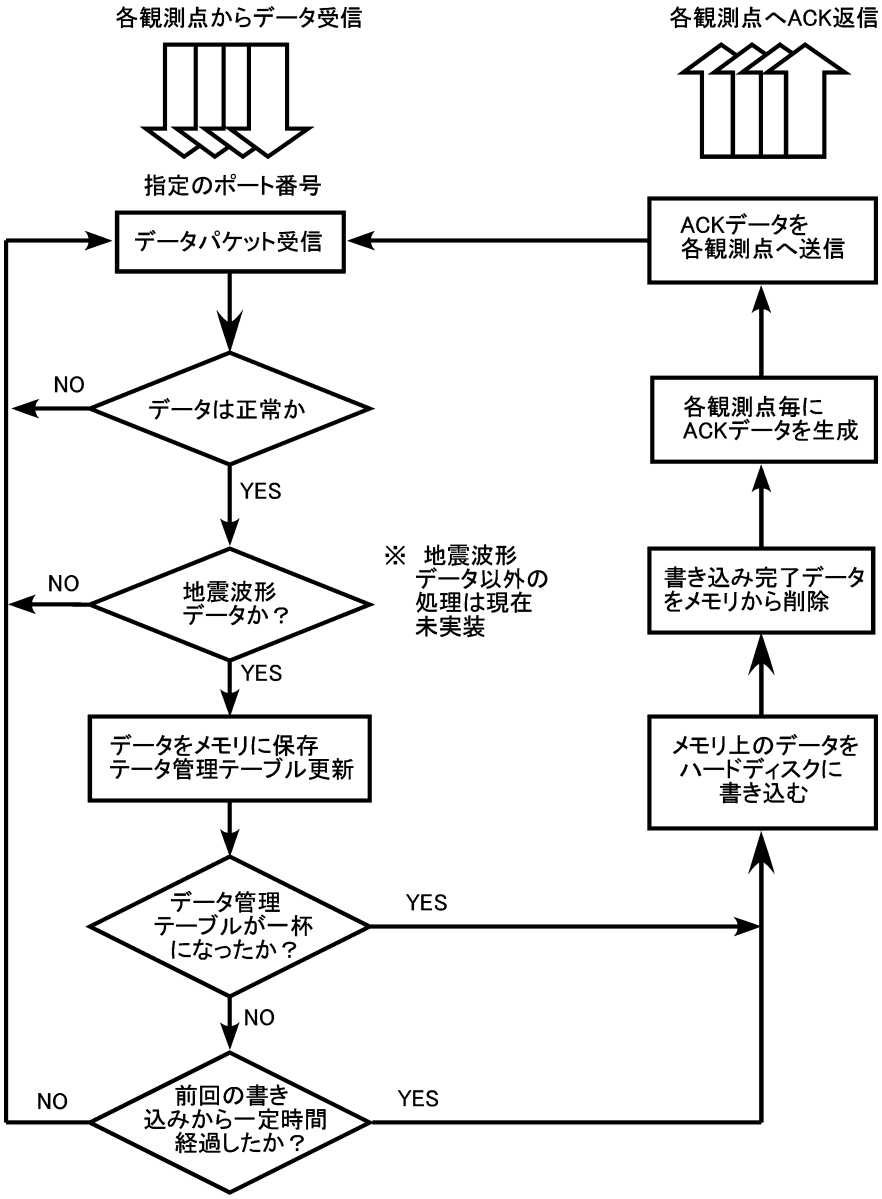

Fig. 2. Flow chart data-receiving system using ACT protocol.

10）送信装置は ACK 信号から 2089（2082），2090，2091（2088）番のデータを消去し，これにより，2080～2090番までのすべてデータが送信されたことになる．

このような手順で，観測点からのデータを確実に，かつ安定して受信装置に送信することができ，データの欠落をなくすことができる．また，ACK 情報をまとめて返信することで，受信装置の負荷はある程度緩和される．

3. 機能の検証

これまで述べてきたように，ACT プロトコルは伝送経路でのデータの欠落があったとしても，それを完全に回復する機能を有している．また，伝送路の混雑や受信装置の過負荷で，正常に送信されるデータの比率が低下した際には，観測点側の判断でデータ送出率を低減し，観測ネットワーク及び観測システム全体が，効率的にかつ安定して機能するように，観測点が自律的に協調する機能を有している．この機能が，考案したとおり動作するか否かを，図 4 で示す方法で検証した．

観測点側装置（送信装置）から，データパケットを伝送路シミュレーターに送信する．伝送路シミュレーターは，設定された妨害率で送信装置から受信したパケットをランダムに破棄し，残りのパケットをデータセンター側装置（受信装置）に送る．受信装置は，送信装置から受信したデータ番号をチェックし，ACK 信号を伝送路

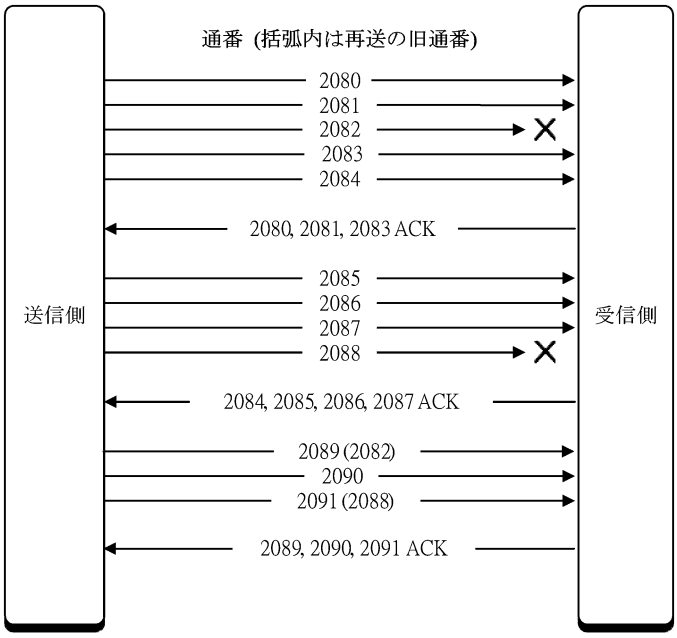

Fig. 3.　Example of packet transmission based on ACT protocol.

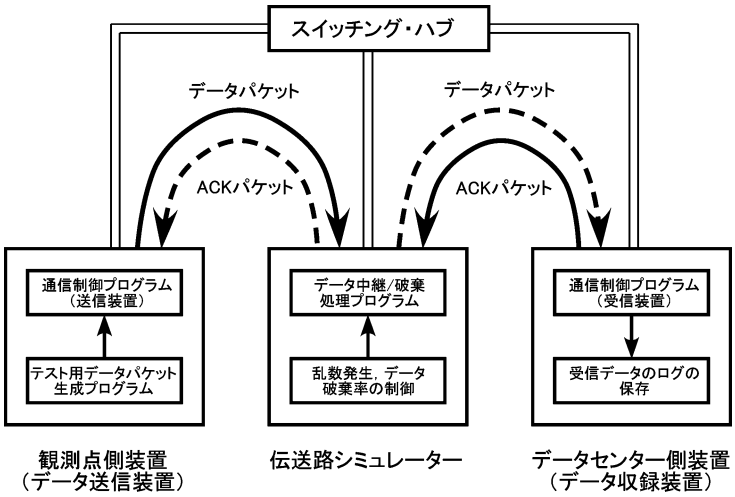

Fig. 4.　Block diagram of packet loss test for ACT protocol.

シミュレーターに返信する．伝送路シミュレーターは，データパケットと同様に ACK 信号も決められた妨害率でランダムに破棄し，残りを送信装置に送る．これにより，上り下りともにデータパケットの遺失がある回線を再現している．

この実験で，送信装置が自律的にデータ送信レートを制御する様子を図 5 に示した．実験は 11 時頃より 14 時頃までの 3 時間行い，図 5（a）のように妨害率を階段状に変化させ，それによって送信レートがどのように変わるかを図 5（b）に，また未送信のデータが送信装置にどのように蓄積されるかを図 5（c）に示した．例えば，12 時 05 分から 15 分までと，35 分から 45 分の各 10 分間は，妨害率を 100% と設定して全てのデータを伝送路シミュレーターが棄却している．このときには，当然ながら ACK 信号も全く届かず，ACK 受信率が 0% になり，生成されたデータは全て送信装置のメモリに蓄積される．ACK 受信率の低下に伴い，自律協調機能によりデータ送信率も下げられる．伝送路シミュレーターの妨害プログラムを停止させ，妨害率を 0% にした 12 時 15 分，及び 45 分以降は，送信装置から受信装置にデータパケットが正常に伝送されるので，わずかな時間遅れの後に ACK 受信率が 100% まで回復する．これにより送信

レートの制限値（*Nx*）が大きくなり，実効的なデータ送信レートも急激に大きくなる．データ伝送レートがデータ生成レートを上回り，メモリに蓄積されていた未送信データも徐々に送信され，徐々に減少してゆく．最後には，メモリ上のデータが無くなり，実効のデータ伝送レートはデータ生成レートと一致し，リアルタイムデータ伝送に近づく．リアルタイムに戻った時点で，観測点で生成されたデータは，すべて欠落なくデータセンターに送られている．

1 つの送信装置に対して 1 つの受信装置の実験では，ネットワークとして正常に機能するかどうかわからない．そのため，複数の観測点から 1 つの受信装置にデータパケットを送信し，それぞれの観測点で妨害率が独立に設定される場合についても実験した． その結果を図 6 に示す．図では 2 つの観測点の状態を上下にして示した．ここでは，上図の観測点を A 点，下図の観測点を B 点と呼ぶこととする．A 点では 14 時 10 分に妨害率を 50% に上げた．この時，データ欠落が急激に発生するためデータの再送が増え，しばらく（1〜2 分間）は，実効データ送信レートが上昇する．しかし，データ欠落が大きくて ACK 受信率が下がることから， 送信レート制限値が下がり，実効送信レートも制限値に張り付いて下がってゆく．この状態では，データ生成にデータ伝送が追いつかないため，データは A 点の送信装置内のメモリに蓄積されてゆく．一方，B 点では，この時点では生成データはほぼリアルタイムで受信装置に送られ，データはメモリには蓄積されていない．14 時 15 分に B 点も妨害率を 50% に上けると，A 点のときと同様に振る舞い， データはメモリに蓄積されてゆく． 14 時 20 分に両点共に妨害率を 10%まで下げると，両点ともメモリに蓄積されていたデータが徐々に送り始められる．この時，どちらかの観測点が急激に送信レートを上昇させることなく，両点とも同じようにゆっくりと蓄積されたデータを減らしてゆく．メモリに蓄積されたデータのすくなかった B 点で，蓄積データの送信が完了すると，回線容量に余裕ができるため A 点の蓄積データが急速に送信され，最後には両点とも蓄積データがメモリからなくなり，リアルタイム伝送に近づく．

上記のように， ここで考案した ACT プロトコルは，回線の状況や受信装置の状況に応じて，観測点の判断で送信レートをダイナミックに変化させ，観測点どうしが協調することにより安定してデータ伝送を行うことが検証できた．この手順の採用により，多数の観測点からの地震データを確実に，かつ安定して取得できるシステムを構築することができた．

4. 運用時の安定性

首都直下地震・減災特別プロジェクトの首都圏地震観測網は， 2009 年 10 月現在 178 観測点が稼動している．狭い首都圏に多数の観測点を設置する必要から，観測点の環境が最適になるように選ぶことは困難で，様々な設置環境が混在している．人口密集地である首都圏は，地動ノイズが大きいだけでなく，電磁輻射ノイズも極めて大きい．例えば，埼玉県川口市にある川口東中学校の校庭に設置された観測点は，近くにあるラジオ送信局からの放送電波により ISDN 回線に非常に大きな電気的なノイズが混入している．そのためインターネット回線が極めて不安定で，データ伝送レートが 1 日の間で大きく変化する．図 7 では，2009 年 4 月 29 日から 5 月 1 日の 3 日間における 1 時間あたりの伝送されたデータパケット数を示している．比較のため，回線状況の良い東京大学地震研究所 1 号館脇に設置された弥生観測点の伝送パケット数も示す．回線状況の良い弥生観測点では， 1 時間に生成されたデータパケットがほぼリアルタイムで確実に伝送されるので，3 日間を通じて 1 時間当たり 3600 パケットで一定である．ところが川口東中学校観測点では，1 時間当たり 400 パケットから 9000 パケットまで変化する．回線状態が悪い時には伝送パケット数が極端に減少し，回線状況が良くなる深夜から未明にかけて観測点に蓄積されていたデータを高いデータ送出レートで伝送してほぼ実時間に回復する．図 7 からこの動作を繰り返していることが判る．ここで利用しているインターネット回線は帯域保証をしていないため，通信会社はこれほど不安定であっても規格の範囲内であるとして，回線品質の改善の要求を受け付けてくれなかった．このような場合には，従来の伝送方式では，データの欠落が頻発して，データ伝送が非常に困難であると思われる．帯域保証のある高品質のインターネット回線を利用することで問題を解決できるが，回線利用料は高くなる．ACT プロトコルは堅牢なデータ伝送手順で，確実に安定してデータ伝送する仕組みである．従って，ここで示したような極めて品質の悪い回線であっても，伝送遅延は非常に大きくなるが，最終的に全ての観測データがデータセンターで受信されている．図 8 は，図 7 で示した期間において， 川口東中学校観測点から伝送されたデータの 1 時間毎の最大遅延時間と平均遅延時間を示している．現在はデータは実時間に近いものを優先して送信するように設定されている．そのため，回線状況が改善して受信装置に蓄積されたデータが多数送信される深夜の時間帯では，遅延時間は最大 90000 秒（約 25 時間）と非常に大きくなる．一方，回線状況が悪い昼間は，データ送信が

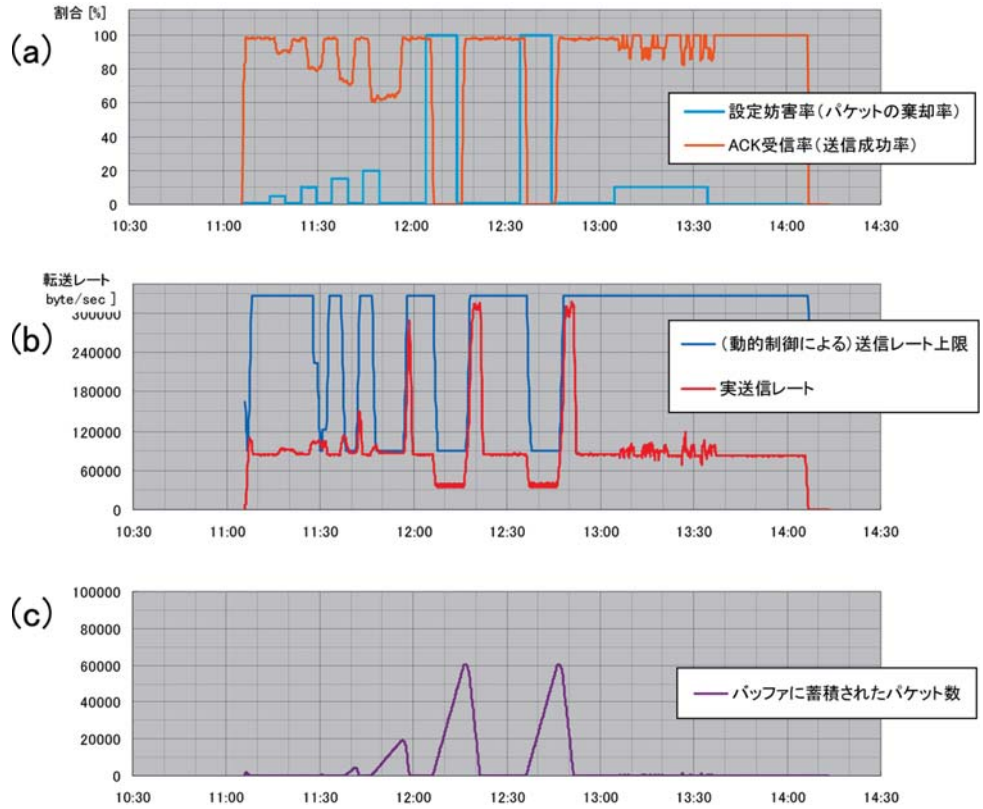

Fig. 5. Result packet loss test for ACT protocol.
(a) Time variation of given obstruction rate (green) and ACK return rate (orange). The ACK return rate is defined as total sent packets and those successfully received.
(b) Time variation of upper limit transmission rate (blue) and effective transmission rate (red). ACT protocol controls the upper limit of transmission rate well, and it is synchronized with ACK return rate in Fig. 5 (a).
(c) Time variation of the amount of data stored in the data sending system. If the data transmission rate is smaller than data acquisition rate, some data cannot be sent to the data center and are is stored in the system.

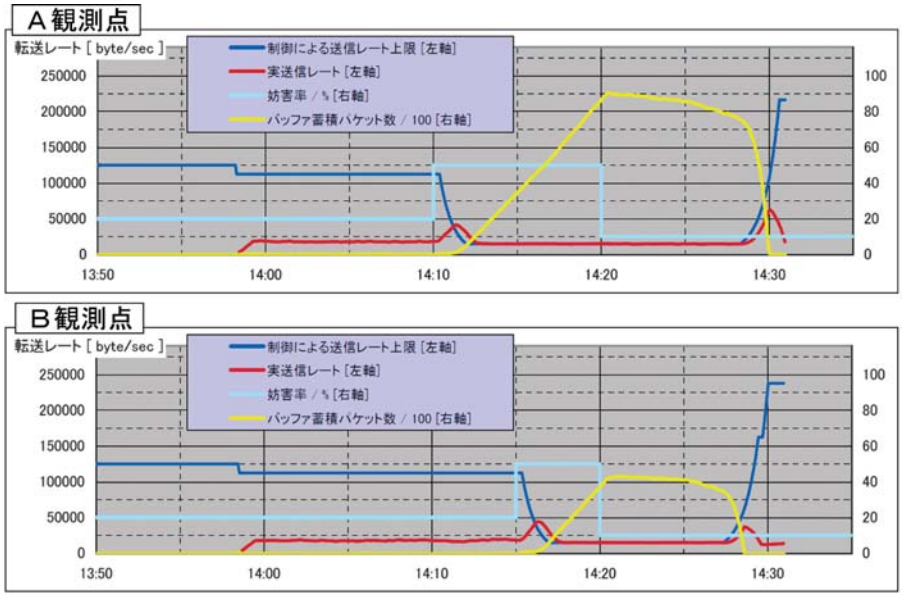

Fig. 6. Time evolution of status for two data-sending systems (upper and lower).
Blue: Upper limit of data transmission rate, Red: Effective data transmission rate, Green: Obstruction rate given for the test, Yellow: Amount of data stored in data-sending system.

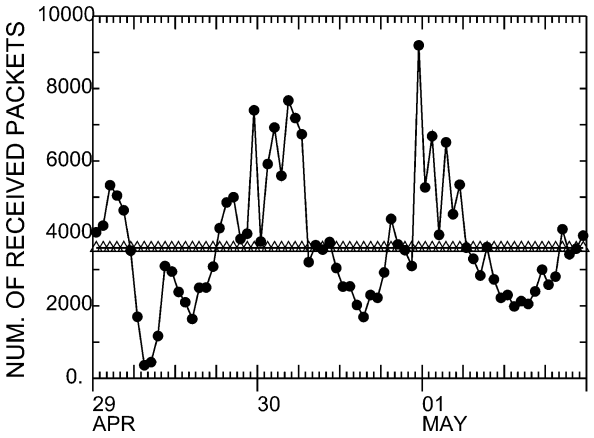

Fig. 7. Hourly number of data packets received from Yayoi station (blue) and Kawaguchi-Higashi-Chu station (red). Yayoi station is situated under good data transmission circumstances and conditions are stable. The hourly flow of data packets is constant. On the other hand, Kawaguchi-Higashi-Chu station operates under poor conditions condition, and hourly volume varies by time. Even under poor conditions, all data are successfully transmitted to the data center. No loss of data packets has been detected.

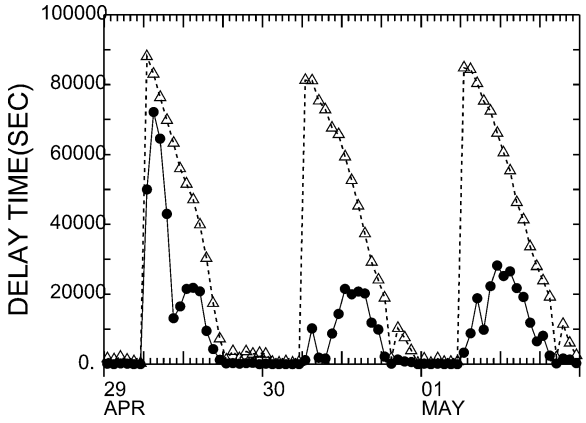

Fig. 8. Maximum delay time (blue) and mean one (red) sent from Kawaguchi-Higashi-Chu station shown in Fig. 7. Even if the delay time is very large all data are successfully sent to the data center using ACT protocol.

あまり成功しないが，成功した場合には直近に取得されたデータであるため，データ遅延時間は小さくなる．奇しくもこの観測点で，品質の悪いインターネット回線であっても確実にデータ伝送ができる実績を示せ，この伝送方式が通信事情の極めて悪い観測点でも利用できることが実証できた．蛇足ながら，伝送遅延を少なくするため，最近この観測点の ISDN 回線に我々自身がノイズ除去回路を付加した．これにより回線品質はかなり向上し，データの伝送遅延は全体として小さくなった．

5. 最後に

本報告では，首都直下地震・減災特別プロジェクトの首都圏地震観測網（MeSO-net）で実用されている新しい世代の地震データ伝送方式である自律協調型データ伝送方式（ACT プロトコル）について説明した．この ACT プロトコルはデータ伝送方式のひとつであり，これを利用して地震波形以外のデータも支障なく送信することができる．現在の MeSO-net 地震観測点は 178 点で，3 成分の地震波形が 200 Hz のサンプリング周波数で伝送されている．1 日のデータ総量は 30 GB となる．それらを全く欠落なく 1 年以上も記録できている．この観測網を運用はじめてからデータの欠落が一切ないだけでなく，受信装置に大きな障害が発生していないことから，この伝送方式の安定性，堅牢性及び確実性は明らかになった．

著者らは，このデータ伝送システムが，我々のプロジェクトを越えて広く普及することにより，データ伝送技術の水準の向上を願っている．それが固体地球物理学の研究分野における我々の貢献のひとつであると考えている．本論では，このプロトコルのデータフォーマットと共に，動作原理を詳しく説明した．更に，C 言語で記述された受信装置のソースコードを以下の URL にて公開し，商用で利用する場合を除き，誰でも無償で利用できるようにする予定である．

日本の地震観測データ伝送技術は，20 年前まではインターネット通信をいち早く取り入れ，世界の最先端であった．ここでは詳述しないが，現在諸外国で利用されているデータ伝送方式は確実性や安定性に優れ，日本方式はかなり見劣りするものとなっている．今回，敢えて本論を記し，ソースコードを公開したのは，単にこの ACT 方式の利用を勧めるという目的ではない．これを契機に ACT 方式以外の新たな伝送方式の出現も含めて，日本の地震観測データ伝送技術が，再び世界の最先端になり，研究分野の発展を望むからである．本論が，地震観測技術の向上の一助となれば幸甚である．

ACT プロトコルの受信プログラム・ソースコード公開サイト

http : / / www. eri. u-tokyo. ac. jp / shuto / report / hokokusho.html

引用文献

平田　直・酒井慎一・佐藤比呂志・佐竹健治・纐纈一起（2009），「首都直下地震防災・減災特別プロジェクト」サブプロジェ

クト①「首都圏周辺でのプレート構造調査，震源断層モデル等の構築等」の概要，地震研究所彙報，**84**，41–56.
石原　靖・幸　良樹・坪井誠司・水谷宏光・深尾良夫・森田裕一・綿田辰吾・竹内希・横山景一・山田功夫・渋谷拓郎（2004），新世代海半球ネットワークの構築，日本地震学会講演予稿集，**No 2**，157.
笠原敬司・酒井慎一・森田裕一・平田　直・鶴岡　弘・中川茂樹・楠城一嘉・小原一成（2009），首都圏地震観測網（MeSO-net）の展開，地震研究所彙報，**84**，71–88.
中川茂樹・鶴岡　弘・川北優子・酒井慎一・平田　直（2009），首都圏地震観測網データセンターの構築と運用，地震研究所彙報，**84**，107–114.
森田裕一・西村太志（1993），GPS 受信機基板を用いた地震観測用高精度時計の製作，地震（2），**46**，67–83.
森田裕一・大湊隆雄（2005），火山における地震観測の発展と成果，火山，**50**，S77–S100.
酒井慎一・平田　直（2009），首都圏地震観測網の設置計画，地震研究所彙報，**84**，57–69.
Uehira, K. (2009) Development of a distributed backup system and a recovery system for telemetric seismic data, Earth Planet Space, **61**, 285–289.
卜部　卓（1994），多チャンネル地震波形データのための共通フォーマットの提案，日本地震学会講演予稿集，**No 2**，384.（その他の情報は http : / / eoc.eri.u-tokyo.ac.jp / WIN / index.html を参照）

(Received November 13, 2009)
(Accepted December 3, 2009)

Appendix

The source code: the main part of data receiving process.

```
// This code is written for explaining the process to receive data packets using ACT protocol,
// but for using in real case. Only main parts are shown here, and some parts in the programs of
// the distributed versions are abbreviated in this list to help understanding the flow of the program.
// Anyone who wants to use ACT protocol should download the source code from the following URL.
// http://www.eri.u-tokyo.ac.jp/shuto/report/hokokusho.html
// All rights are reserved.
//
// データ管理テーブルの構造
// 観測点-i から受信した n 番目のパケットは、pb[cl[i].list[n-1].indx_pbrec]でアクセスできる
struct rel_cl_pb { long long  seq; int indx_pbrec; int result_code; };
// 受信パケット管理情報
struct clinfo {
    unsigned char setflag;                      // このレコードに client が登録済のとき 1
    unsigned long addr;                         // client の IP アドレス
    unsigned short port;                        // client の送信(返信宛先)ポート
    struct rel_cl_pb list[ MAX_NUMPB_PER_CL];   // この client から受信したパケットの、
                                                //パケットバッファとの対照表
                                                //
    int cur_indx_per_cl;                        //上記構造体配列の indx を追跡
    long long cur_max_seq;                      //上記構造体に記録した sequence の現時点最大値
                                                //(書込完了時、ACK 生成に使用する)
    long long cur_min_seq;                      // 上記構造体に記録した sequence の現時点最小値
                                                //(書込完了時、ACK 生成に使用する)
    long long ackseq;                           // ACK パケットの通番
};

ChSetupBox *chsetupbox;
static int sigflag=FALSE;

/************************************ Main program ************************************/

int main(int argc,char **argv)
{
    struct recvconf cf;
    int i ;

    /* コマンド・オプションを解析する */
    getoption(argc,argv,&cf);
    /* シグナル処理 */
    signal( SIGTERM, sig_close) ;
    signal( SIGINT,  sig_close) ;
    signal( SIGPIPE, SIG_IGN ) ;
    /* チャンネル情報の格納領域確保と読み込み */
    chsetupbox = ( ChSetupBox*)malloc( sizeof( ChSetupBox));
    if ( readchconf( chsetupbox, &cf) ==-1 ) {
        fprintf(stderr,"fail to read the channel file.¥n");
        return 1;
    }
```

```
    /* 受信ループ　通常は常時このプログラム（orderrecv）が動作している*/
    orderrecv(&cf);
    /* 終了処理 */
    for ( i = 0 ; i < RELTYPE_MAX ; i ++ ) {
        ui_udpclose(cf.reludp[ i]) ;
    }
    exit(1);
}

static void sig_close(int sig)
{
    /* ループを抜けてプログラムを終了するための変数のセット */
    sigflag=TRUE;
}

/********************** データ受信プログラム本体 ************************************/

static void orderrecv(struct recvconf *cp)
{
    UDP *src ;
    ORDEREDPACKET order ;
    char buf[ 5000] ;
    char *plbuf, *tmpstr = NULL ;
    long long seq ;
    fd_set fds ;
    struct timeval tv ;
    int sock, retval, srclen, payloadlen ;
    int i, j, indx_cl = 0, cur_indx_pbrec ;
    struct clinfo cl[ MAX_NUM_CLIENTS] ;  /* 観測点情報, ACK 管理テーブル */
                    /* MAX_NUM_CLIENTS はこのプログラムが処理できる最大の観測点数 */
    struct pacbuf pb[ MAX_NUM_PBRECORD] ;  /* データ管理テーブル */
    void *ptr ;
    long clock_per_sec ;
    clock_t clock, pclock ;
    int flag_do_write ;
    int pb_max, cpb_max, pkt_stat ;

    /* UDP ポートを開く */
    src=udpopen(NULL,cp->srcport,TRUE,TRUE);
    if (src==NULL) {
        perror("udpopen");
        return;
    }
    udpsetbuffer(src,65536*8);
    sock=udpgetsocket(src);

    if ( cp->pacbufmax==0 ) {
        pb_max=MAX_NUM_PBRECORD;
    }
    else {
        pb_max=min(cp->pacbufmax,MAX_NUM_PBRECORD);
    }
    cpb_max=min(MAX_NUMPB_PER_CL, pb_max);
```

```
for (i=0;i<MAX_NUM_CLIENTS;i++) {
    cl[i].setflag=FALSE;
}
for (i=0;i<MAX_NUM_PBRECORD;i++){
    pb[i].ppl=NULL;
}

cur_indx_pbrec=0;
flag_do_write=FALSE;
clock_per_sec=sysconf(_SC_CLK_TCK);
pclock=times(NULL);

while (1) {
again:
    FD_ZERO(&fds);
    FD_SET(sock,&fds);
    tv.tv_sec=0;
    tv.tv_usec=50000;
    retval=select(sock+1,&fds,NULL,NULL,&tv);
    if (retval==-1) {
        if (errno==EINTR)
            goto again;
        perror("select");
        return;
    }

    if ( retval!=0 && FD_ISSET(sock,&fds) ) {

        srclen = udprecv( src, buf, sizeof(buf)) ;

        /* 受信パケット内容の検査 */
        pkt_stat = chk_received_pkt( buf, srclen) ;

        for ( i = 0 ; i < MAX_NUM_CLIENTS ; i ++ ) {
            if ( cl[ i].setflag == FALSE ) {
                /* 新規クライアントを登録 */
                cl[ i].setflag = TRUE;
                cl[ i].addr = src->addr;
                cl[ i].port = ntohs(src->dport);
                cl[ i].ackseq = 0LL;
                cl[ i].cur_indx_per_cl = 0;
                cl[ i].cur_max_seq = cl[ i].cur_min_seq = 0LL;
                // the array should be terminated by a record of list[].index_pbrec==-1
                for ( j = 0 ; j < MAX_NUMPB_PER_CL ; j ++ ) {
                    cl[ i].list[ j].indx_pbrec = -1 ;
                }
                break;
            }
            else if ( cl[ i].addr == src->addr  &&  cl[ i].port == ntohs( src->dport) ) {
                /* 登録済クライアントに一致 */
                break;
            }
        }
        if ( i<MAX_NUM_CLIENTS ) {
```

```
        /* クライアント数が上限を越えていない場合 */
        indx_cl = i;    /* 格納 */ }
    else {
        pkt_stat |= CHKRP_TOO_MANY_CL ;
    }

    /* ACT パケットの要素を取り出す */
    memcpy( &order, buf, offsetof( ORDEREDPACKET, mark)) ;
    payloadlen = chtowd( order.payloadlen) ;
    seq = chtollwd( order.seq) ;
    plbuf = buf + offsetof( ORDEREDPACKET, mark);
    tmpstr = plbuf + offsetof( struct winftmt, datetime);

    if (      ( pkt_stat & CHKRP_AO_NOT1SECxCH) != 0
          ||  ( pkt_stat & CHKRP_AO_NO_CHLIST)  != 0
          ||  ( pkt_stat & CHKRP_AO_F_INCONSIS) != 0
          ||  ( pkt_stat & CHKRP_AO_BAD_TS)     != 0
          ||  ( pkt_stat & CHKRP_AO_LEAPSEC)    != 0 ) {
        /* AO だが、書込を行わないパケット */
        irreg_ao_write( &fp_irreg_out, cp, ( unsigned char *)plbuf, payloadlen) ;
        if ( ( pkt_stat & CHKRP_TOO_MANY_CL) == 0 ) {
            /* クライアントが登録されていれば ACK を管理する */
            flag_do_write = clbuf_register( cl+indx_cl, seq, pkt_stat, cpb_max) ;
    }  else {  /* AO、書込  */
        if ( ( pkt_stat & CHKRP_TOO_MANY_CL) == 0 ) {
            /* 書込用バッファに登録する */
            if ( (ptr = malloc( payloadlen)) == NULL ) {
                perror( "malloc");
                assert( ptr != NULL);
            }
            /* pb にペイロードをセット */
            memcpy( ptr, plbuf, payloadlen);
            pb[ cur_indx_pbrec].ppl = (unsigned char *)ptr;
            pb[ cur_indx_pbrec].pllen = payloadlen;
            cl[ indx_cl].list[ cl[ indx_cl].cur_indx_per_cl].indx_pbrec = cur_indx_pbrec ;

            /* インデックスのインクリメント */
            cur_indx_pbrec ++;
            if ( cur_indx_pbrec >= pb_max ) flag_do_write = TRUE ;

            /* クライアントが登録されていれば ACK を管理する */
            if ( clbuf_register( cl+indx_cl, seq, pkt_stat, cpb_max) ) {
                flag_do_write = TRUE ;
            }
        }
    }
}       /* 受信処理の終了 */

clock = times( NULL) ;
/* 書き込みフラグが立つか一定時間が経過の場合, データをハードディスクに書込み */
if (    flag_do_write
    ||  clock - pclock >= ( clock_per_sec*( cp->tout_msec) / 1000) ) {

    pclock = clock;
```

```
            /* データをハードディスクに書き込む */
            hdwrite( pb, cur_indx_pbrec, &rep, cp);

            /* ＡＣＫを返信する */
            return_ack( *src, cl, pb, cp) ;

            /* パラメータの初期化 */
            flag_do_write = FALSE;
            init_pb_structs( pb, cur_indx_pbrec);
            cur_indx_pbrec = 0;
            for ( i =0 ; i < MAX_NUM_CLIENTS ; i ++ ) {
                if ( cl[ i].setflag == FALSE ) break;
                for ( j = 0 ; j < MAX_NUM_PBRECORD; j ++) {
                    if ( cl[ i].list[ j].indx_pbrec == -1 ) break ;
                    cl[ i].list[ j].result_code = 0 ;
                    cl[ i].list[ j].seq = 0LL;
                    cl[ i].list[ j].indx_pbrec = -1 ;
                }
                cl[ i].cur_indx_per_cl = 0;
                cl[ i].cur_max_seq = cl[ i].cur_min_seq = 0LL;
            }

            if ( fp_irreg_out != NULL ) {
                if ( fclose( fp_irreg_out) != 0 ) perror("fclose") ;
                fp_irreg_out = NULL ;
            }

            if ( sigflag != FALSE ) {
                break;
            }
        }

    }    /* end of while loop */

    return;
}
```