第 15 部

NTP

第 1 章

はじめに

現在広く普及している、ユーザと計算機が端末を介して対話的に作業を進める計算機環境では、計算機はその内部に時計を持ち現在時刻を保持しているのが普通で、たとえばユーザからの求めに応じて現在の日時を提示したり、ファイルシステムにアクセスが発生した時刻を記録する場合を始めさまざまな局面で利用されている。

計算機がネットワークに接続されておらず単独で稼働している場合には時計の精度が問われることは少なく、日常生活において利用される腕時計や置き時計と同程度の精度があれば十分で、10 秒程度の遅れ進みは問題にならない。例外は計算機を利用した計測制御システム中で時刻情報が参照される場合であるが、これも基準時刻からの経過時間が正確に計れれば済む場合が多く、時刻そのものの精度が秒以下オーダで問われることは少ない。

しかし、計算機がネットワークに接続され、リモートファイルシステムや分散処理が普及すると計算機間の時刻のずれが問題となり、秒以下の精度で時刻が一致していないと的確な処理が行なえない場合がでてくる。たとえば複数の計算機がネットワークを介して同一のファイルシステムをアクセスした場合、全ての計算機の時刻が一致していないとファイルアクセスの順序を正確に把握できなくなるといった問題が生じる。このため計算機同士がネットワークを介して時刻を同期させる技術が必要になる。

さらに計算機ネットワーク同士が相互接続されるインターネット環境では、ネットワーク間の時刻のずれが問題となる。ネットワーク間でファイルシステムを共有したり分散処理を行なうことは今のところあまり一般的でないが、将来それが可能になるとネットワーク間で時刻が同期していないことは大きな問題となる。この問題の対策としては、正確な時刻を保持する計算機をネットワーク上のどこかに用意し、他の計算機はこの計算機に時刻を同期させる方法が考えられる。そのための手法のひとつが ntp (network time protocol) [171] であり、現在インターネット上の多くの計算機がこのプロトコルにもとづいて時刻同期を行いミリ秒の精度で互いの時計の時刻を一致させている。

NTP ワーキンググループは、文部省国立天文台の研究者をメンバーに迎え 1992 年度に新たに発足したワーキンググループで、WIDE プロジェクトでは数少ない共同研究グループである。本ワーキンググループは、国立天文台が管理運用するセシウム原子時計を時刻情報源とする計算機を準備し、この計算機の時刻情報をインターネットに安定供給することを目標に活動を開始した。

本報告は 8 章からなり、その一部は国立天文台のメンバーが執筆した。具体的には、1

章を大野 (WIDE) が、2 章を福島 (NAO [1]) が、3 章を大野が、4 章を大野、鈴木 (WIDE)、久保、松田 (以上 NAO) が、5 章を福島、松田、鈴木が、6 章を鈴木が、7 章を福島が、そして 8 章を大野が執筆した。

[1] 国立天文台の略称

第 2 章

UTC と 国立天文台の役割

現在、我々が日常生活に用いている時刻系を協定世界時 UTC (CoordinatedUniversal Time [1]) という。

UTC は、世界中の原子時計から構成される国際原子時 TAI(International Atomic Time) と、地球の自転運動の観測から決定される世界時 UT1 (Universal Time 1) の二つの時刻系を、以下のように組み合わせて作られている。まず UTC の時間単位、つまり 1 秒の長さは TAI の時間単位である国際原子秒と同じであると決められており、国際原子秒は平均海水面 (厳密にはジオイド) 上で計った国際単位系 (SI) における 1 秒、すなわちセシウム 133 のある特定の放射の周期の 9192631770 倍の時間と定義されている。UTC の 1 日は、日出没や太陽の南中などの天文現象とは一切関係なく、機械的に 86400 国際原子秒である。

一方、年月日時分秒を示す UTC の時刻自体は UT1 からあまり離れないように調整されている。TAI は、1958 年の初めに世界時と一致するように決められたが、地球の自転が月などの潮汐力に基づく摩擦によりわずかであるが遅くなってきているため、その後は世界時とは全く独立に維持されてきている。具体的な調整法は、UT1 が TAI に比べて遅れぎみであるため、余分の一秒 (閏秒という) をほぼ年に一回挿入することにより UT1 と UTC の差の絶対値が 0.9 秒を越えないようにしている。この結果、UTC と TAI とは 1 秒の整数倍しか違わない。1993 年 4 月現在、TAI と UTC の差は TAI−UTC=27 秒である。これまでの TAI と UTC の差の経過を表 2.1 に示す。

TAI は、世界中の天文台や度量衡の研究所あるいは時刻通報にたずさわる機関などに設置された多くの原子時計を相互に比較し、重み付き平均をとることにより決定されている (表 2.2 参照)。残念ながら、現在大陸間の時計比較法の主流となっている人工衛星を使った方法では、比較精度が数十ナノ秒と個々の原子時計の測定精度より 1 桁以上悪いので、TAI ひいては UTC そのものの精度はあまり良くない。このため、精度の点だけでいえば個々の研究所でのローカルな原子時計の方が高精度である。したがって、ローカルな原子時計に基づく協定世界時も重要となってくる。このとき、研究所の略符号を付加してローカルな UTC を表す。例えば、国立天文台 (三鷹) の場合、天文台で維持している UTC は UTC (NAOT) と表現する。これに対して、略符号が付かない UTC は theUTC、すなわち平均として決定される TAI と直接結び付けられた真の UTC を指す。TAI およ

[1] これは UTC に限らないが、時刻に関する略語のスペルの順序が変っているのは、原語がフランス語だからである。

表 2.1: TAI と UTC の関係

年	月	日	ΔT
1972	1	1	10
1972	7	1	11
1973	1	1	12
1974	1	1	13
1975	1	1	14
1976	1	1	15
1977	1	1	16
1978	1	1	17
1979	1	1	18
1980	1	1	19

年	月	日	ΔT
1981	7	1	20
1982	7	1	21
1983	7	1	22
1985	7	1	23
1988	1	1	24
1990	1	1	25
1991	1	1	26
1992	7	1	27
1993	7	1	28

注: ΔT は TAI − UTC を秒単位で表した量である。

び UTC は決定精度こそ高くはないが、全世界的な時計比較を行うことにより時刻系としての確度は個々のローカルな原子時や協定世界時に比べて格段に向上しているといえる。

さて、ローカルな UTC は真の UTC より精度的には良いことが多いので、必ずしも値が一致するようには運用されておらず、1 マイクロ秒程度の差はしばしば生じている。これは、オフセットや一定の進み遅れはハードウェア的にあるいはソフトウェア的に補正できるのであまり気にしていないという「時計屋」の感覚によるものである。むしろ 2 次以上の高次ドリフトとかワウやフラッターなど短周期的な揺らぎを抑制するよう運用するのが常である。このことは、ローカルな UTC をマイクロ秒の精度で利用する場合には気を付けていなければならない。

TAI と UTC の管理は、他の計量単位であるメートルやキログラムと同じく、1875 年に締結されたメートル条約に基づき設置された政府間委員会である国際度量衡総会 CGPM およびその執行機関である国際度量衡委員会 C IPM が行っているが、実際の運用は CIPM の直接の下部組織である国際度量衡局 BIPM(パリ) において行われている。日本では、主に文部省国立天文台と郵政省通信総合研究所が国際原子時の維持に貢献している。特に、国立天文台は全世界をカバーするための 3 つの主要ノードの一つ (アジアノード) であるため、TAI および UTC の維持に枢要な位置を占めている。国立天文台では、11 台のセシウム原子時計を用いてこの任に当たっている。(図 2.1)

一方、閏秒を挿入するかどうかは VLBI(超長期線電波干渉計) ほかの天文観測によって得られる UT1 の動きに基づき国際地球回転観測事業 IERS (Internat ional Earth Rotation Service) とよばれる国際機関によって判断される。閏秒の挿入は、いつもというわけではなく、12 月末もしくは 6 月末に行われる。例えば 6 月末の場合、6 月 30 日 23 時 59 分 59 秒と 7 月 1 日 0 時 0 分 0 秒の間に 6 月 30 日 23 時 59 分 60 秒という閏秒が挿入される。挿入は UTC の 0 時、すなわち日本時の 9 時に行われる。

表 2.2: TAI および UTC に寄与している原子時計群

国	研究所	略記号	所在地	数	寄与点	Δ UTC
アメリカ	米海軍天文台	USNO	Washington	51	1319	0.030
フランス		F		20	969	−0.504
ドイツ	連邦物理工学研究所	PTB	Braunschweig	8	687	2.840
アメリカ	国立標準技術研究所	NIST	Boulder	10	615	−0.134
スイス		CH		11	445	−0.250
ロシア	国立物理電波技術計測研究所	SU	Moscow	4	400	0.328
日本	通信総合研究所	CRL	小金井	8	375	2.529
オーストラリア		AUS		7	353	0.030
イギリス	国立物理研究所	NPL	Teddington	7	311	0.374
日本	国立天文台三鷹	NAOT	三鷹	6	231	−1.281
オーストリア	グラーツ工科大学	TUG	Graz	3	202	1.263
日本	国立天文台水沢	NAOM	水沢	2	200	−0.534
スペイン	アルマダ物理研究所	ROA	San Fernando	6	186	3.024
イタリア	国立電子技術研究所	IEN	Torino	3	147	−0.522
台湾	電気通信研究所	TL	Chung-Li	3	134	−0.100
オランダ	ヴァン・スインデン研究所	VSL	Delft	4	130	−0.089
日本	計量研究所	NRLM	つくば	3	103	−16.683
中国	峡西天文台	CSAO	Lintong	4	87	−0.547
ドイツ	応用測地学研究所	IFAG	Wettzell	4	62	2.983
スウェーデン	国立時刻周波数研究所	SNT	Stockholm	3	58	0.079
中国	上海天文台	SO	Shanghai	2	46	2.110
ポーランド	国立標準研究所	PKNM	Warsaw	3	38	0.092
カナダ	国立科学評議会	NRC	Ottawa	2	33	1.477
イタリア	カリアリ天文台	CAO	Cagliari	3	28	−28.762
アメリカ	応用物理研究所	APL	Laurel	5	26	−0.036
イスラエル	国立物理研究所	INPL	Jerusalem	3	11	−0.698

注: 表は、現在 TAI もしくは UTC に寄与している世界中の研究所を、研究所毎の寄与点の順に並べたものである。スペースの関係で、寄与点が 10 以下の研究所 (19 個所) は割愛した。ただし、フランス、スイス、オーストラリアについては国内すべての研究所をひとつとして扱っている。寄与点は、個々の時計について過去一年間の安定度 (主に時間に線形なドリフトを除いた後の分散) の状況により BIPM(国際度量衡局) がつける評価点 (最高 100 点) を研究所別に加えたものである。NAOT(国立天文台三鷹) と NAOM(国立天文台水沢) のように、同じ機関であっても所在地が異なる場合は別の研究所として計上している。表中の寄与点は 1993 年 2 月 21 日現在のもので、この時点での総寄与点は 7298 点である。Δ UTC は、研究所のローカルな UTC と真の UTC との差 UTC(local) − UTC をマイクロ秒単位で表した量である。

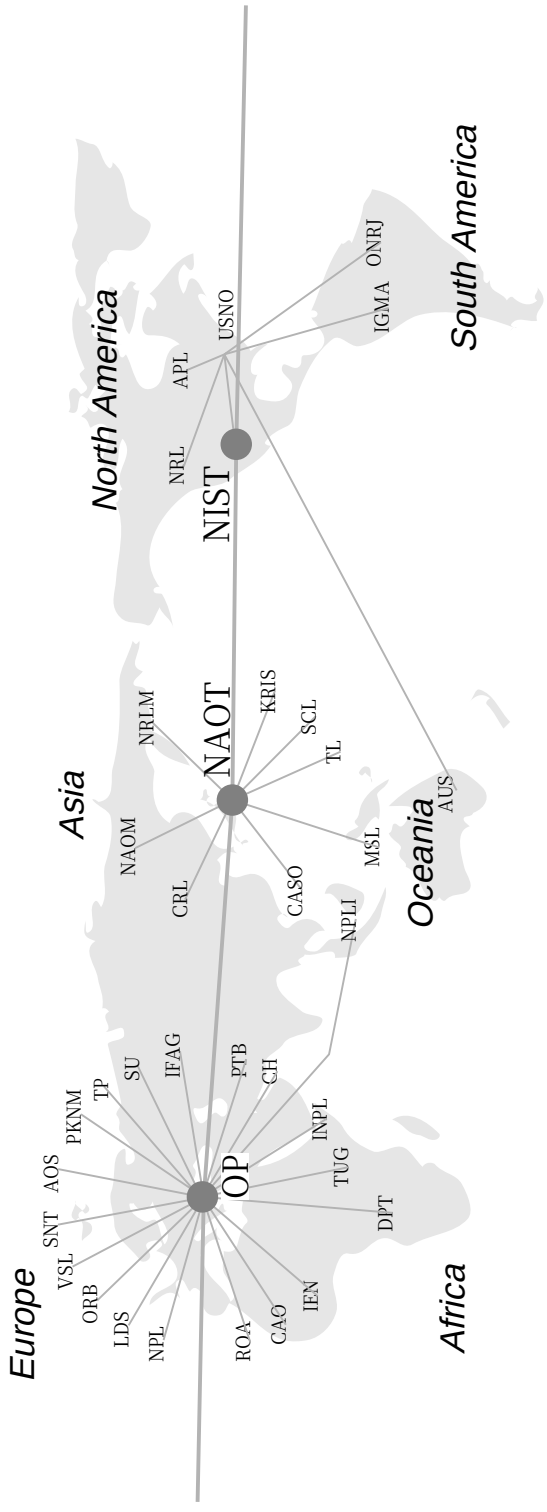

図 2.1: セシウム原子時計の分布

第 3 章

ntp の概要

3.1 ntp による時刻同期

ネットワーク上の計算機が時刻同期を行なう方法として、rdate コマンド、timed などが開発され利用されてきたが、現在最も普及している方式は ntp [171] である。ntp が普及したのは rdate や timed が以下に挙げる問題を抱えていたためである。

rdate:

- 指定した計算機の時刻に強制的に合わせてしまうので、修正される側の計算機では時刻が不連続に変化する。
- 時刻情報を伝送する際の遅延時間を考慮していない。

timed:

- ブロードキャスト通信を用いるのでネットワークの形態によっては利用できない。
- timed を稼働している計算機すべてが「合議制」によって現在時刻を決定するため、決定された時刻が正確な時刻であるとは限らない。
- 時刻情報を伝送する際の遅延時間を考慮していない。

上記の問題点を回避し、大規模なネットワーク上で正確な時刻情報を共有する機構としてデラウェア大学のミルズ (David L.Mills) によって開発されたのが ntp で、上記の 2 方法と異なり UTC にミリ秒の誤差で同期させることができる。ntp の特徴は以下の 2 点を前提として設計されていることである。

- ネットワーク上に正確な時刻情報を保持する計算機がある。
- ある計算機と通信を行なう際、行きと帰りの伝送遅延時間が等しい。

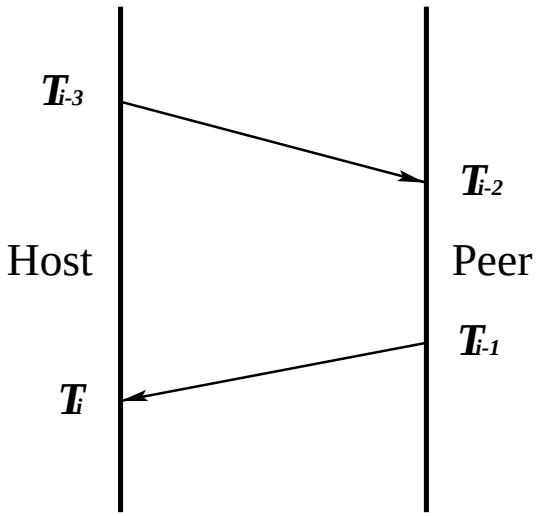

図 3.1: ntp パケットの交換

ntp では、2 つの計算機がネットワークを介して情報交換をし、一方の時計を他方に同期させる。このとき以下のような方法で通信路の遅延を評価する (図 3.1)。

図 3.1は 2 台の計算機 Host と Peer が ntp パケットを交換する際のタイムチャートで垂直方向上から下へ時間が流れているものとする。このとき右側の Host の時計を左側の Peer の時計と一致させるには、ある時刻 T'において Host と Peer 双方の時計が示す時刻の差 $\Delta T = T'_{Peer} - T'_{Host}$ を求める必要がある。 Host 側で正確に把握できるのは、Peer に向けてパケットを送った時刻 T_{i-3} と Peer からパケットが送られてきた T_i のみであり、Peer が Host からのパケットを受けとった時刻 T_{i-2} やパケットを Host に送り出した時刻 T_{i-2} は伝送遅延が未知である以上直接知ることはできない。そこで、Peer から Host に返送するパケットに T_{i-1} と T_{i-2} を付加する。 T_{i-1} と T_{i-2} がわかり、伝送遅延が行きと帰りで等しいという仮定が成り立つ時 ΔT は以下の式で求められる。

$$\Delta T = T_{Peer} - T_{Host} = \frac{T_{i-1} + T_{i-2}}{2} - \frac{T_i + T_{i-3}}{2}$$

ntp プロトコルを実装したプログラムはいくつか発表されているが、ntp の最新版 (version 3) に対応しているプログラムとしては xntp とよばれるパッケージが著名である。このパッケージには、ntp プロトコルを処理するデーモンである xntpd 以外にいくつかの支援プログラム (ntpdate、ntpq、 xntpdc など) が含まれている。xntpd は、内部で (1) パケット受信 (Receive Process)、(2) パケット送信 (Transmit Process)、(3) 時刻更新 (Update Process)、(4) 時刻調整 (Local Clock Process) の 4 つの処理を行なっている。これらの関係を図 3.2 に示す。

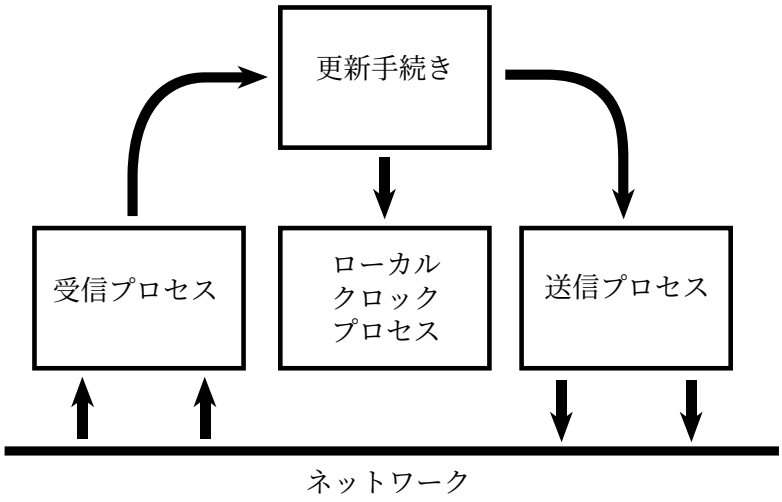

図 3.2: ntp のモデル

3.2 階層 (stratum) の導入

前節でネットワーク上に接続されている他の計算機の時刻と自計算機の時刻のずれ (ΔT) を算出できることを示した。次に必要なのは正確な時刻情報の入手である。

ワークステーションを始めとするほとんど全ての計算機の内部時計の基準発信器は水晶発振子を用いている。水晶発振子を用いた時計は日常生活においては十分な精度を提供するがミリ秒の精度で時計を常に一致させるためには精度が足りない。もし水晶発振子より精度の高い信号源が入手できても、時刻を UTC などに一致させる方法が提供されていなければ全ての計算機の時刻がある一定値以内に収まるに過ぎない。

そこで、すでに ntp が稼働しているインターネットでは、ネットワーク上に UTC を供給する専用の機材を含んだサーバを何台か設置し他の計算機はこのサーバを参照するようにしている。ただし他の全ての計算機がこのサーバを直接参照すると、このサーバやネットワークに不要な負荷がかかるので、階層 (stratum) という概念を導入し UTC を保持するサーバを頂点 (stratum 1 という) とする階層構造を構成している。すなわち ntp を利用する全てのサーバはいづれかの stratum に属し、時刻調整は自分と同じ stratum に属するサーバか、自分より 1 階層だけ頂点に近い（すなわち、より精度の高い情報を保持していると考えられる） stratum に属するサーバとだけ行なう。この方法には以下の特徴がある。

- stratum 1 サーバの精度がネットワーク上の全てのサーバおよびそれを参照している計算機の精度に影響を与える。
- stratum 数の小さいサーバほど stratum 1 サーバとの誤差が少ない。

したがって、各サーバの stratum 数はネットワークにかかる負荷を考慮した上で可能な限り小さくするのが普通である。

理想的には各組織に最低 1 台は stratum 1 サーバが設置されるのが好ましいが、日本の現状は理想から大きく隔たっている。

米国には stratum 1 サーバを運用する組織が何箇所か存在するため、これらを参照する stratum 2 サーバも数多く存在するが、1992 年度初頭の日本国内には stratum 1 サーバは設置されておらず、WIDE インターネットを始めとするいくつかのネットワークで米国の stratum 1 サーバを参照する stratum 2 サーバを運用しているにすぎない。そして WIDE インターネットなどに接続されている各組織は国内最高位の stratum 2 サーバを参照しているため、各組織の stratum は最高でも 3 であり、他のサーバの stratum は 4 ないし 6 となっている。

第 4 章

セシウム原子時計に基づく NTP stratum 1 サーバの試作

すでに述べたように、1992 年度初頭の段階では日本国内に stratum 1 サーバは存在しておらず、正確な時刻情報 (UTC) の供給はもっぱら米国の stratum 1 サーバに依存していた。一方、国内で ntp を採用する組織は増加を続けており、これらの組織が独立して米国の stratum 1 サーバをアクセスすると不必要な負荷をネットワークにかけると予想された。そこで WIDE プロジェクトでは WIDE インターネットの NOC (Network Operation Center) に stratum 2 サーバを設置し、各組織にはこの stratum 2 サーバを参照するよう提案したが、この方式には各組織の stratum が最良でも 3 になるという欠点がある。このため国内に stratum 1 サーバを新設し、各組織はこれを参照して stratum 2 サーバ確保したいという要望が高まってきた。

そこで、WIDE プロジェクトの大野と鈴木は文部省国立天文台の福島らとの情報交換を 1992 年度初頭から開始し、同天文台が管理運用するセシウム原子時計を時刻情報源とする stratum 1 サーバの実現可能性について検討を始めた。その結果、以下の理由から本ワーキンググループを発足させ本格的な研究活動を開始することになった。

- 技術的に特に大きな障害が見あたらない。
- 天文台には必要な機材が整備されており、それらを利用できる。
- 国内に stratum 1 サーバを設置することが国内のインターネットの発展に貢献する。
- 世界的に見てもセシウム原子時計を時刻情報源とする NTP サーバに関する報告がなされていなかった。

本章では、本ワーキンググループが試作したセシウム原子時計を時刻情報源とする stratum 1 サーバについて述べる。

4.1 設計方針

セシウム原子時計に基づく ntp サーバーの構成にはさまざまな案 [172] があったが、現段階では以下のような方式を採用している。これを Mk–I 方式と呼んでいる。

1. セシウム原子時計から正秒信号と 1MHz の正弦波信号をとりだす。
2. これを 2 組の 32 ビットバイナリカウンタに送り計数する。
3. カウンタの出力を、パラレル入力ボードを介してパソコン (PC9801) に取り込む。
4. このパソコン上には ntp サーバのサブセットを用意しておく。
5. 光ファイバで外界と電気的に遮断したイーサネットにこのパソコンを接続し、対外サービスを行う。

この方式を採用した理由は、以下のとおり。

1. 国立天文台のセシウム原子時計から出力される時刻信号には、UTC (厳密には UTC (NAOT)) に正確に同期した秒パルス信号および 1MHz、5MHz、10MHz の正弦波信号があるが、このうち 1MHz の正弦波信号が ntp で取り扱うデータタイプの一つ (tv_usec) の計数単位と一致し、利用しやすかった。
2. 国立天文台では、秒信号および 1MHz の正弦波信号を常時利用可能で、かつ 2 つの時刻信号の同期がマイクロ秒以下の精度で保証されていることから、単純な 2 進積算回路で正確な計数ができると予想されたこと。(伝送エラーがありうる場合の時刻積算回路では、10 進化回路、相互比較による判断回路、あるいは平滑化・統計回路などが必須であるが、これらを省略しても、高速なレスポンスが期待されたこと)
3. 内部同期精度の目標をマイクロ秒としたため、高速型 (遅れ 500 ナノ秒以下) のパラレル入力ボードが必要となったが、国内で容易に調達可能なものは PC9801 用しかなかったこと。
4. 当初は、時刻信号を取り込んだパソコンとは別に ntp サーバを搭載したワークステーションを用意し、相互に通信しあいながら他の組織に対して時刻情報を提供する方針であったが、サーバとして想定していた SONY NEWS の内部時計分解能 (粒度) が 10 ミリ秒と大きく[1] 、また、ntp サーバプログラムが部分的に移植が可能になったため、信号取り込み機能と ntp サーバ機能を一体化した方がよりよい結果が得られると予測されたこと。
5. 落雷によるサージや急激な電圧低下などの電気的な不安定要因がイーサネットを通じて時刻信号経路を逆行し、供給元の UTC 信号、さらに原子時計側に影響する事態が予想さため、システム全体を無停電電源でバックアップし、さらに天文台内の他のネットワークとの間を光ファイバーで結んで、他のネットワークとの間を電気的に分離するするのが好ましいと思われたこと。

ここでは上記の 2)、3)、4) について順次検討する。

[1] 後述「ワークステーション内部時計分解能比較表」を参照。

4.2 ハードウェア — ntp サーバ用カウント回路

計算機ネットワーク上において 1 マイクロ秒の精度で計算機相互間の時刻同期を実現するため ntp サーバ用カウント回路の設計と製作を行なった。

製作した回路は、UTC に同期した原子時計の 1pps 信号及び 1MHz 信号を入力とし、これをもとに 64 ビットの時刻データに変換して、計算機のパラレル入力ボードに送出するものである。

ハードウェアの設計を担当した著者の一人は、原子時計を扱うものの立場から、当初時刻データとして 1 秒信号及び 1MHz 信号を計数するなら 10 進カウンタを利用するのが妥当だと考えた。しかし、計算機内部での信号処理は 16 進数に変換した方が好都合なので、カウンタ部分にはポピュラーな同期式 4 ビットバイナリカウンタを 8 個カスケードに接続して、 32 ビットのカウンタを形成した[2]。

次に、原子時計と計算機ネットワークが電気的に相互干渉を起こさないように、フォトカプラによって、カウント回路出力部を絶縁することを検討した。しかし、標準的フォトカプラの応答速度は目的とする 1 μ s の分解能とくらべて遅く、各々のビットがそれぞれが異なった遅れを持つ可能性がある。そこで、出力部ではなく入力部をパルストランスで絶縁することにした。この方式ならば原子時計の信号出力 (1pps 信号:10Vp-p、パルス巾 20 μ s、立ち上がり時間 50ns 以下/1MHz 信号:1Vrms) に十分対応できる。

全体の回路構成を図 4.1に示す。入力部は 1pps 信号及び 1MHz 信号共にパルストランスで絶縁され、アッテネータを介して次段へ接続されている。 このうち 1pps 信号は分配アンプにより 2 分配されており、一方は同期式 4 ビットバイナリカウンタ 8 個で形成される 32 ビットの秒カウンタへ入り、もう一方はワンショットマルチバイブレータ (Single Shot) へ入る。 一方、1MHz 信号は ゼロクロスコンパレータによってサイン波から矩形波に変換され、秒カウンタと同じ構成の 32 ビットのマイクロ秒カウンタへ入る。 ここで 1pps 信号を分配したのは、マイクロ秒カウンタを 1 秒信号でリセットすれば秒以下の部分がマイクロ秒単位の 16 進数 32 ビット (実際は 20 ビット) で表わされるためである。

また、マイクロ秒カウンタをリセットする信号は、分解能 1 μ s を得るために、ワンショットマルチバイブレータでパルス巾約 500ns の信号に変換している。マイクロ秒カウンタは 32 ビット全てをカウントするか秒信号でリセットするかをスイッチによって選択でき、秒カウンタは データロード スイッチによって任意の値にプリセットできる一方、リセットスイッチによってゼロにクリアすることもできる。なお、32 ビットカウンタがオーバーフローするにはマイクロ秒カウンタで約 71.5 分を要し、秒カウンタでは 約 136.1 年を要する。通常はマイクロ秒カウンタを 1pps 信号でリセットするようにしておけばマイクロ秒カウンタのオーバーフローは気にしなくともよい。

試作の段階では、当初、秒カウンタ出力部及びマイクロ秒カウンタ出力部にバッファ機能がなかったため、ケーブルを延長した際に反射が生じて計数が正常に行なわれないという障害があった。そこで、出力段にバッファアンプを設けカウンタと計算機との間を結ぶケーブルを 1 メートル程度まで延長できるようにした。

[2] ここで、32 ビットのカウンタを 8 ビットカウンタの 4 段接続にしなかった事に特に意味はない

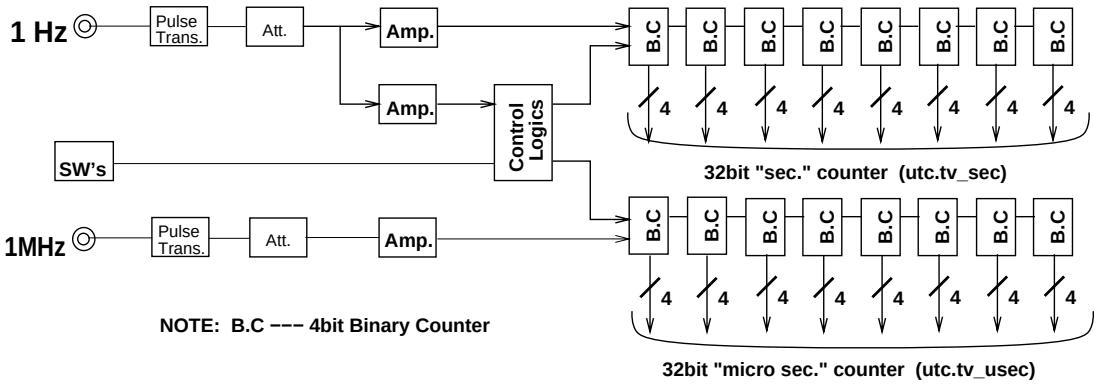

図 4.1: カウンタ装置の構成

以上が原子時計の 1pps 信号及び 1MHz 信号をそれぞれ 32 ビットのバイナリデータに変換するハードウェアの概要である。この ntp サーバ用カウント回路は 2 台試作した。1 号機は AC100V を電源としたが、2 号機は AC100V 電源に加えて DC 電源での運転も可能にし、さらに時刻を表示するデジタル時計回路と 2 つの 32 ビットカウンタの動作をモニタする LED 回路を追加した。(図 4.2参照)。

4.3　パーソナルコンピュータ用ソフトの開発

パーソナルコンピュータ側のソフトは、stratum 1 となるワークステーションにセシウム原子時計の時刻情報を転送する機能を持つ。

開発言語は、ワークステーションとの互換性を考えて C 言語とし、マイクロソフト C Ver.6 とそれに対応したアンガマンバスのソケットライブラリを使用した。

セシウム原子時計の 1pps および 1MHz 信号は、前節で述べたカウンタで 32 ビットの秒およびマイクロ秒単位で計数される。このうち秒カウンタは、UNIX で標準的に採用されている構造体 struct timeval の tv_sec [3]に一致させることができるので、一度設定すればソフトウェアによる修正や変換は不要である。なお、原子時計で ntp サーバを実現する場合には、上に述べたとおりマイクロ秒カウンタを 1pps 信号でリセットするように出来るので、マイクロ秒カウンタが 1000000(1 秒) 以上の値になることを考慮する必要がない。そのため、マイクロ秒カウンタ 32 ビットのうち上位 12 ビット ($2^{20} = 1000000\mu$秒) はコントロールビットなどに転用可能である。

パソコン上のソフトウェアは、パラレルインタフェースボードから 8 つの I/O アドレスに割り当てられた 8 ビットのポートを介してデータを読み込む。こうして得られた 8 つの 8 ビットデータを以下のような構造体 struct timeval に格納する。

```
struct timeval {
```

[3]1970 年の 1 月 1 日 UT 0 時からの積算秒を現している

```
        long  tv_sec;       /* seconds */
        long  tv_usec;      /* microseconds */
    }
```

この struct timeval 型の構造体に格納された時刻情報は、ソケットライブラリを使ったデータ転送プログラムにより他の計算機に転送できる。

なお、カウント回路の動作確認のために上記プログラムとは別にパラレルインタフェースボードで読み込んだ 64 ビットデータをモニターするプログラムも併せて作成した。このモニタープログラムはカウンタ装置から送られてくる 64 ビット分のデータを、8 ビット毎に 2 進および 16 進表示するもので、ハードウェアや struct timeval 取得ルーチンのデバッグ時に効果的であった。

図 4.2: カウンタ装置の外観

第 5 章

NTP サーバーに関する時計比較実験

Stratum 1 サーバーを実際に運用するには、ntp サーバーの基本的部分、すなわちネットワーク上の異なる計算機の内部時計同士の時刻比較をどれぐらいの精度で行えるかを知る必要がある。このために、前の章で述べた方式に基づき、計算機内部でセシウム原子時計に基づく時刻信号を実現しているパソコン (以下、簡単のために PC と呼ぶ) とシステム時計のみを参照する普通のワークステーション (以下、簡単のため、WS と呼ぶ) との間でイーサネットを介した時計比較実験を行った。

実験方法は、PC をクライアント、WS をサーバーとし、おのおのの時刻系で測った時刻を往復通信でやり取りすることにより、回線の遅れを (一応) 差し引いた時計差を計測するというものである。以下では、時刻についてクライアントでは小文字を使って t_1、t_2 などと表し、サーバーでは大文字を使って T_1、T_2などと表すことにする。

実験に使用した計算機などの諸元は以下の通りである。

PC (クライアント)		
	ハードウェア	NEC 社製 PC9801DA
		(CPU i80386、クロック 20Mhz)
	OS	MS-DOS Ver.3.3C
	C コンパイラ	Microsoft C Ver.6
	ライブラリ	アンガマンバスソケットライブラリ
WS (サーバー)		
	ハードウェア	SONY 社製 NWS821
		(CPU M68020、クロック 16.67Mhz)
	OS	NEWS-OS Ver.3.3C
	使用システムコール	gettimeofday

実験環境は、国立天文台三鷹地区のメインのネットワークにつながっているワークステーションと、メインネット下のサブネットにつながっている PC を用いた。メインネットワーク及びサブネット内で閉じたトラヒックはかなりの量であるが、二つのネットワークをまたがるトラヒックは比較的少ない状況であった。時計比較に使ったプログラムの概要は以下の通りである (図 5.1)。

クライアント (PC) では、

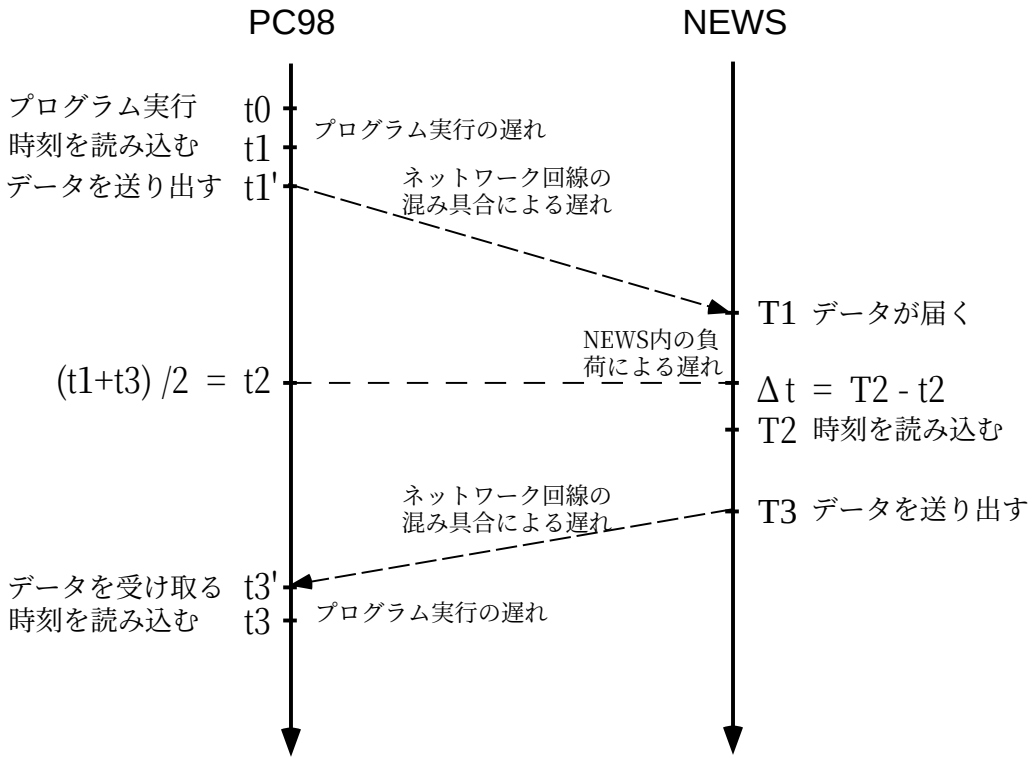

図 5.1: 経過時間に沿ったソフトウェアの動作

1. 計測開始の合図をうける。
2. クライアントの内部時計を読みだす (この読みを t_1とする)。
3. 読んだ値を文字列に変換し、サーバーに向けて送信する。

サーバー (WS) では、

1. クライアントの通信を受信する。
2. サーバーの内部時計を読みだす (この読みを T_2とする)。
3. 読んだ値を文字列に変換し、クライアントに向けて返信する。

これをうけて、クライアントでは、

1. サーバーからの返信を受信する。
2. 再びクライアントの内部時計を読む (この読みを t_3とする)。

さて、T_2に対応するクライアント側の時刻を t_2とする。回線の遅れが行きと帰りで一定であり、計算機内部の時間遅れを無視できるとすると

$$t_2 = \frac{(t_1 + t_3)}{2}$$

としてよいであろう。そこで、WS の時刻 T_2との時刻差 Δt を

$$\Delta t = T_2 - t_2$$

と定義し、これを連続して多数回 (例えば 100 回) 測定する。もちろん、実装に当たっては、高速性を確保するために測定結果はすべて配列としてメモリ上に保存するなどの工夫が必要であるが、ここではその詳細には立ち入らない。

ところで、現実には、さまざまな要因により上に述べた仮定は正しくない。例えば、クライアントもサーバーも時計を読む時刻と通信データを送受信した時刻は異なる。パソコンではシングルタスクで動いているので、この時刻のずれは送受信や時計読みだしなどの行為のプログラムのステップ数に比例しており、パソコンの動作クロックが安定である限りほぼ一定であるとみなしてよい。一方、ワークステーションではマルチタスクであるので必ずしも一定とは言い切れない。実際、ユーザプロセス実行中は常に割込みや DMA が起きる可能性がある。しかし、多数回測定することにより、これらの例外的なイベントは少数例として取り除くことができ、少数のそのような例外を除去したのち平均操作を行えば確度の高い時計差が得られるであろう。

以下に実験結果を述べる。まず、一回の測定に要する時間は、回線の混み具合にもよるがおおよそ 2 ミリ秒であった。測定結果の一例を時間順にプロットしたものを図 5.2に示す。

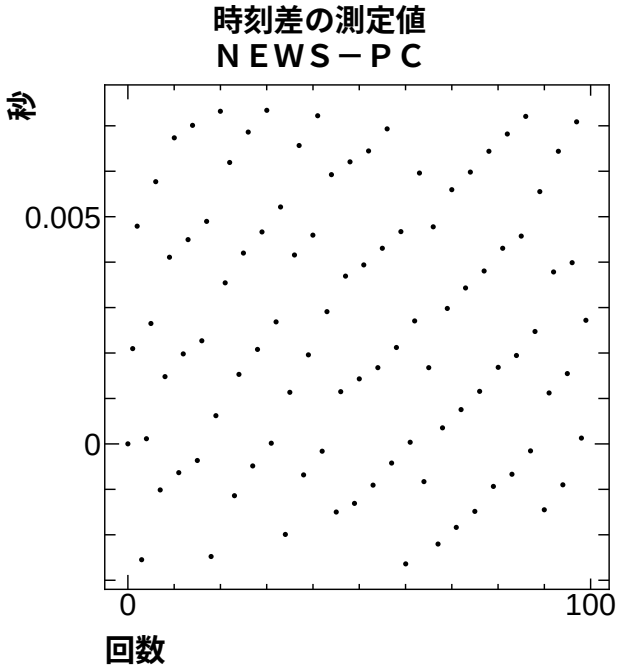

図 5.2: NEWS-PC 間における時刻差 (時間順)

図 5.3: NEWS-PC 間の時刻差 (頻度分布)

横軸は何番目に測定したかのインデックス (1 から 100 まで)、縦軸は測定値Δt である。この場合、時計差の平均値は 1 回の 100 回連続測定で標準偏差 3 ミリ秒程度で決定できる。これをさらに多数回繰り返すことによって時計差の平均値のばらつきは約 1 ミリ秒の精度で決定できることがわかった。

しかしながら、これはこのシステムの限界ではない。もう一度図 5.2を見てみよう。一見、測定値は十分バラついているように見えるが、注意深く見ると測定値が鋸波状に増減を繰り返しているのがわかる。これは、サーバー (WS) の内部時計の分解能 (粒度) が 10 ミリ秒しかないためにおきた現象だとおもわれる。

図 5.3は測定値の頻度分布をとったものであるが、測定値の分布が幅 10 ミリ秒の一様分布に近いことがわかる。実際、測定値に単純な統計処理を施すと、標準偏差が約 3 ミリ秒程度となるが、これは正規化一様分布の分散が 1/12 であることから導かれる理論値

$$\sigma = \frac{10}{\sqrt{12}} = 2.88 \text{ ミリ秒}$$

とよく一致している。これは、測定値のばらつきがサーバーの内部時計の粗さによるためであることを裏付ける有力な証拠である。

さらに、比較のため、同様の実験をセシウム原子時計に基づく時刻信号を実現しているパソコン同士で行なった。どちらのパソコンも同一の時計信号を入力している。クラ

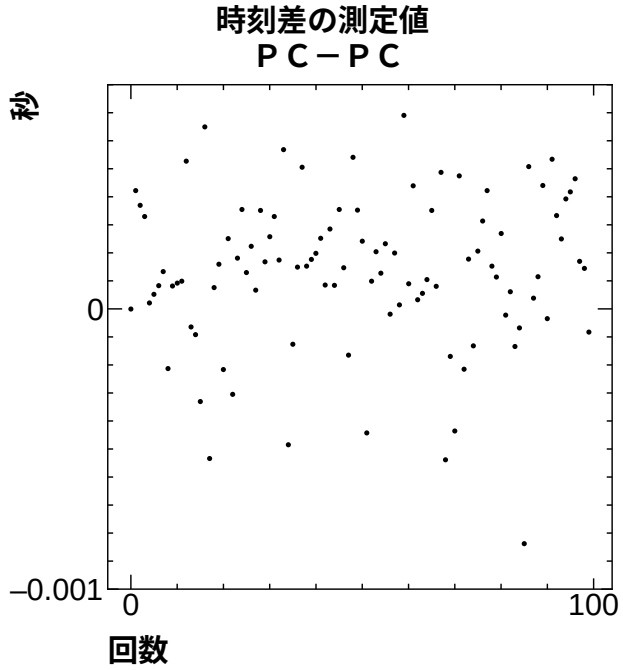

図 5.4: PC-PC 間における時刻差 (時間順)

イアント側のパソコンは上記と同じものを用い、サーバー側のパソコンはハードウェアだけ PC9801RA(CPU i80386、内部クロック 20Mhz) に置きえただけで後はまったく同じである。ネットワークなど実験環境は同じにした。

このときの測定結果の一例を時間順にプロットしたものを図 5.4に、その頻度分布を、図 5.5に示す

PC–PC 間の実験では、サーバーもクライアントもマイクロ秒の分解能を持っているため、測定値の分布はガウス分布でよく近似できる。この場合、時計差の平均値は 1 回の 100 回連続測定で標準偏差 300 マイクロ秒程度で決定でき、これをさらに多数回繰り返すことによって時計差の平均値のばらつきは約数十マイクロ秒の精度で決定できることがわかった。

したがって、この実験結果から同一構内のネットワークすなわち LAN ベースであればわれわれの方式により数十マイクロ秒の時刻同期が可能であると推測される。実際にこの精度で時刻同期を実現するには、高い分解能の内部時計を持つワークステーションを用いて実験を重ねる必要があろう。

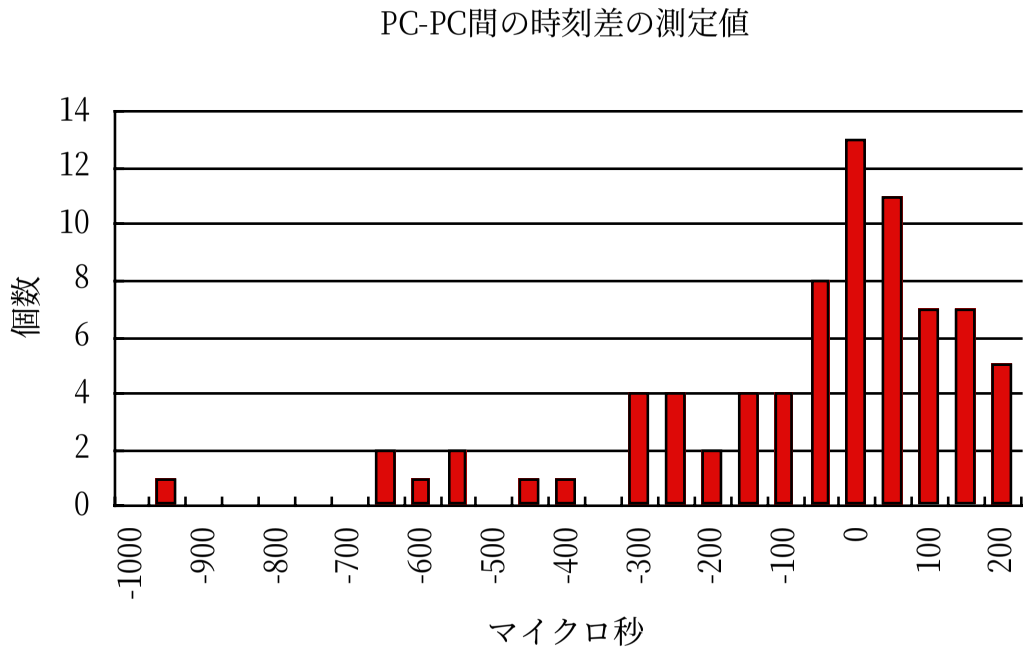

図 5.5: PC-PC 間における時刻差 (頻度分布)

第 6 章

xntpd の修正による暫定 ntp サーバ実験

xntpd を修正することで暫定的に ntp サーバを立ち上げてみる実験を行った。まだ実験中であるためレポートとしては不完全であるが、現状を報告する。

6.1 動機

最終的にどのような形で ntp サーバを実現するのかは、まだ議論の余地があるが、すでに述べたように、いくつかのパーツが現実に動作している。

- セシウムクロック
- バイナリカウンタ
- パラレルインターフェースを実装した PC9801

最終的な ntp サーバに必要な機能のうち現時点で不足しているのは、以下の二つのうちいずれかである。

- PC9801 上で動作する、完全な ntp サーバ
- 何らかの UNIX が動く機種で動作する ntp サーバへのデータの正確な受渡し機構

現実問題として、ntp の仕様を全て満たすサーバを PC9801 上で実装するのには多くの労力と時間がかかる。逆に、UNIX 上でセシウム原子時計を直接利用するには、まだハードウェアが不足している。

そこで、正確さという点では難点があるが、まず実験環境を用意し、正確さについての情報を得るために、イーサネットを介して UNIX 上で動作する xntp[1]サーバと PC9801 上で動作するセシウムクロックカウンタを連係する実験を行った。

本来 ntp サーバは直接接続されているデバイスから読みとるのにどれだけ時間がかかるのかなどといったパラメータを持つことにより、なるべく誤差が生じないように工夫している。そこにイーサネットという全く時間的に不安定な要素を取り込むのは問題であるが、以下の点から有効であると判断した。

[1]実験では、最新版である Version 3 サーバを用いた

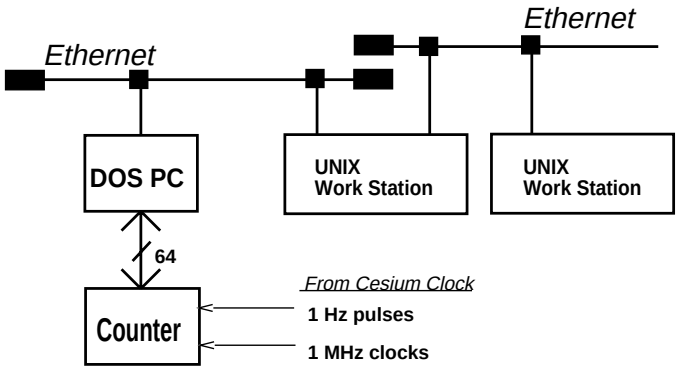

図 6.1: xntp 実験環境

- ntp サーバを動かすために必要な技術を整理できる
- 前の章の実験ではっきりしているように、同じセグメント上に他のイーサネットノードがつながっていなければネットワークトラヒックによっておこる遅延による影響は比較的少ないはずであり、UNIX が本来もつリアルタイム性を阻害する要因を考えれば差はないといえる。
- 反対に、ネットワークを間に介することによってどの程度影響が出るのかを知る手がかりとなる

以下、その方法について説明する。

6.2 実験系の構成 (ハードウェア)

実験系の構成を図 6.1に示す。

PC9801 と実験用ワークステーション (今回の実験は SONY NEWS) を同一イーサネットセグメントに接続する。さらに、観測用のワークステーションを用意した。実験用ワークステーションは Internet 的に接続されているので、他のサイトから実験内容を参照することのみならず ntp サーバとして実際に参照できるため ntp プロトコルを介してクロックを実際に同期させる事が出来る。

6.3 実験系の構成 (ソフトウェア)

ソフトウェアは PC9801 側と NEWS 側双方に用意する。

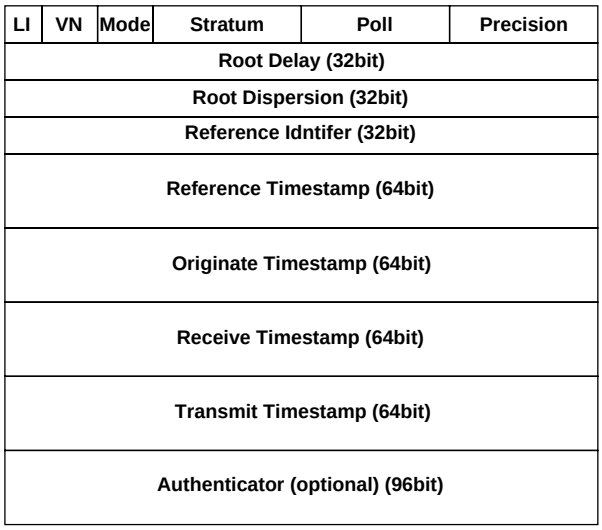

図 6.2: ntp プロトコル パケットフォーマット

6.3.1　PC9801 側のソフトウェア

ntp プロトコルの一部を実装したプログラムを、先に述べたソケットライブラリを用いて作成した。ntp のパケットが送られて来ると、時刻などの情報を埋めこんだリプライパケットを作り返信するというものである。ここで言う「時刻」とは、PC9801 に接続されたバイナリカウンタで設定された値である。

具体的には、ntp packet(パケットフォーマットは図 6.2参照) を受け取ると、まず、受け取った時刻を Receive Timestamp に設定した後、以下の値を設定する。

- Stratum = 0
- Polling Interval = 64 (本来正しくない)
- Precision = 0
- Root Delay = Root Dispersion = 0
- Reference Clock Identifier には ATOM(原子時計) を設定
- Reference TimeStamp には、本来、最後に時計が校正された時刻が入るべきであるが、適当な値が無いため受け取り時刻と同じ時刻をセットする。

この後、パケットを実際に送出する直前に、Transmit Timestamp に時刻をセットし、パケットを送出する。

6.3.2 ワークステーション側ソフトウェア

ワークステーション側は、SONY NEWS に xntpd を若干変更したものを用いる。当初は他のリファレンスクロックと同等の扱いを正しくするようにサブルーチンを記述する予定であったが、ソフトウェア担当グループが xntpd というプログラムの全貌をまだ十分に把握できていなかったため、以下のような方針で修正を行った。

- xntpd を local clock を参照するように設定する (ntp.conf)
- local clock の取り出しは gettimeofday を呼び出すが、その部分で PC9801 側とやりとりを行って値を返すルーチンをはめ込む

6.4 実験

以上のハードウェアを組合せて適切にセットアップを行うと、セシウムクロックを基準とする ntp サーバの準備が整う。このサーバから、観測用のワークステーションがクロックを参照すれば、そのワークステーションはセシウムクロックに正確に同期する。

実験はセシウムクロックを厳密に温度管理しない状態で行ったが、ミリ秒単位までクロックは狂うことはない[2]ので実験に支障はない。

この状態で、外部[3]から来たクロックとの比較を行った。比較は観測用ワークステーションと Internet に接続された他のサイトの双方で行った。

具体的には、xntd は local clock を読み出すように設定されているため、gettimeofday システムコールを呼び出すが、その部分が置き換えられているため、PC 側の簡易 ntp サーバへ問い合わせパケットを送出する。簡易 ntp サーバはセシウムクロックを読み、必要なリプライパケットを組み立てて返送する。xntpd は得られた結果を元に xntpd 内のクロックを同期させる。つまり、UTC(NAOT) に同期するわけである。このことから分かるように、xntpd が動作しているワークステーションのクロックはこの xntpd によっては同期されない。

一方、ワークステーション上の xntpd 以外のプログラムは、正しい gettimeofday システムコールを用いてワークステーションの内蔵時計を参照しているので、この双方のクロックを比較することで稼働状況を知る事が出来るわけである。さらに WIDE インターネット上の何台かのマシンから、国立天文台の xntpd サーバを参照し、時刻情報が取得できることを確認した。

---

[2] 天文台の保時室ではナノ秒のオーダでの管理が行われている。

[3] 正確には国内には stratum 1 サーバはまだ存在しないので、海外の stratum 1 サーバを間接的に参照している。

6.5　今後

1992 年度初頭より開始された本研究は、ハードウェアの構成の検討などから始めて、ようやく ntp サーバが立ち上がる状態に達した。

しかし、実験を始めた直後で、まだ十分な数値的データがとれていないため、客観的な結果を示すには至っていない。

今後の計画としては、ntp プロトコルおよび xntp について理解を深め、「暫定サーバ」にこの知識を反映させる。次に、IBM PC コンパチ機などに直接クロックを読み出す事が出来るハードウェアを用意し、xntpd のドライバを作成する予定である。

第 7 章

今後の展望

当面の目標としては、まず、現在のセシウム原子時計ベースの ntp サーバーを長期間試験運用し、なるべく早い時期に定常運用に移行することが肝要である。定常的に運用する場合、国立天文台側にはなんら問題はなく、むしろ国の共同利用機関として、広く全国の研究者ひいては国民に利用されるサービスを行うことは望ましいとさえいえる。

次のステップとして以下のことが考えられる。

1. 現在のサーバシステムを時刻信号取り込みの部分も含めて PC9801 から DOS/V 系に移植する。
2. 何らかの形でパソコンあるいはワークステーションからハード的に時刻信号 (秒パルス) あるいは位相信号 (例えば 1MHz) をとりだし、計算機内部で想定している時計面と現実の世界での UTC の比較をタイムインターバルカウンターなどでハードウェア的に計測できるようにする。
3. イーサネット以外の各種の伝送経路 (FDDI など) を用いて通信実験を行い、LAN を介してどの程度の測定精度が得られるかを追求する。
4. マイクロ秒の分解能をもつ内部時計を有するワークステーションとの間で通信実験を行い、イーサネット上でどこまでワークステーションの時刻同期が行えるか同期精度の追求を行う。

例えば、1) ができれば諸外国でも同様のサーバーが導入しやすくなるであろうし、2) が確立されればこのサーバーが提供する時刻系の確度が飛躍的に向上する。また、3) や 4) が達成されれば電波干渉計や地震計アレイなど時間差方式を用いた天文学や地球物理学などの観測精度を高めることが可能となり、これらの学問の発展への寄与は大きい。

しかし、ntp を初めとする各種広域時刻同期システムが電話回線 (一般であれ専用回線であれ) を利用する限り、伝送による時刻遅れおよびその揺らぎを解消することは難しい。これを解決するには stratum 1 サーバが数量的にかつ通信体系の上で広範に分布することが重要である。たとえ、日本全体の計算機ネットワークで国立天文台が数十マイクロ秒の精度・確度のサーバを提供したとしても、いったん電話回線が介入すると精度・確度ともに劣化してしまう。したがって、理想的には各 LAN のノードに一台ずつ、ここで述べたような stratum 1 サーバが普及することが望ましい。しかし、セシウム原子時

計は 1 台約一千万円と高価であり、なおかつその精度維持には多大の労力と維持費がかかるため、セシウム原子時計ベースのサーバを普及させることは現実的ではない。

この目的のためには GPS 時に基づく ntp サーバーを開発することが近道であろう。GPS とは全世界測位システム (Global Positioning System) の略で、1990 年代にいつでも世界中どこでも位置と時刻を知る手段として米軍により導入されたシステムである。このシステムでは、約二十個の人工衛星にセシウム原子時計をのせ、TAI と同期された時刻系 (GPS 時) に基づき時刻信号を常時発射させている。歴史的理由により GPS 時は TAI に比べてきっかり 9 秒だけ遅れている。式で書けば以下のようになる。

$$\text{TAI} - \text{GPS 時} = 9 \text{ 秒}$$

GPS 時は TAI と同様 UTC とは関係なく進む時刻系であるが、 UTC との差に関する情報も GPS 衛星から放送されているため、受信機から UTC の時刻情報を取り出すことができる。セシウム原子時計のように秒信号を直接パルスの形で取り出すことは普及版の GPS 受信機ではできないが、一部の専用受信機では可能で専用受信機の場合ほぼマイクロ秒の精度が得られる。この時注意しなければいけないことは、GPS 時は真の UTC に基づいてではなく、米海軍天文台 (USNO) のローカルな UTC によって運用されているという点である。このため、マイクロ秒の確度はないと考えなければならない。ちなみに、TAI の決定に用いるためのローカルな原子時計間の時刻同期は、高級な専用受信機を複数台使って特殊なスケジュールに基づいて観測することにより行われており、その同期精度は数十ナノ秒である。

価格面から見ると普及版の受信機だと 20 万円以下、専用受信機でも 100 万円程度であり、セシウム原子時計に比べはるかに廉価である。さらに GPS 受信機はターンキーシステムでありかつメンテナンスフリーであるため、導入・運用は非常に容易である。さらに心強いことに GPS 受信機はかなりの数の日本メーカーが開発に携わっているため、ntp サーバー用に特注することも不可能ではないと考えられる。

さて、秒パルス信号が出力できる専用 GPS 受信機に基づく ntp サーバ (これを Mark II 方式と呼ぶ) を開発するには、現在の Mark I 方式の大半がそのまま利用でき、変更点は以下に挙げる程度なので、その開発は比較的容易であると予想される (図 7.1 参照)。

1. GPS 受信機は正確な秒パルス信号しか出さないので、時刻積算回路のなかで自分自身でマイクロ秒を積算する必要がある。今のところ、最も有力な手法としては、内部に水晶発振回路を埋め込み、受信機からの秒パルスに同期するよう位相同期ループをかける方法を考えている。

2. GPS 自身がシステムとして正式運用に入った現在、日本上空で GPS 衛星が 1 個も見えないなどの理由で GPS 時の秒信号が途絶することはあまりないだろうが、それでも 24 時間確実に受信できる保証はまったくない。従って、GPS 時の場合、単なる積算秒信号をそのまま tv_sec とするわけにはゆかず、受信機が受信した年月日時分秒などの時刻情報を RS232C 経由で計算機内部に取り込み、きちんと同期させる必要がある。

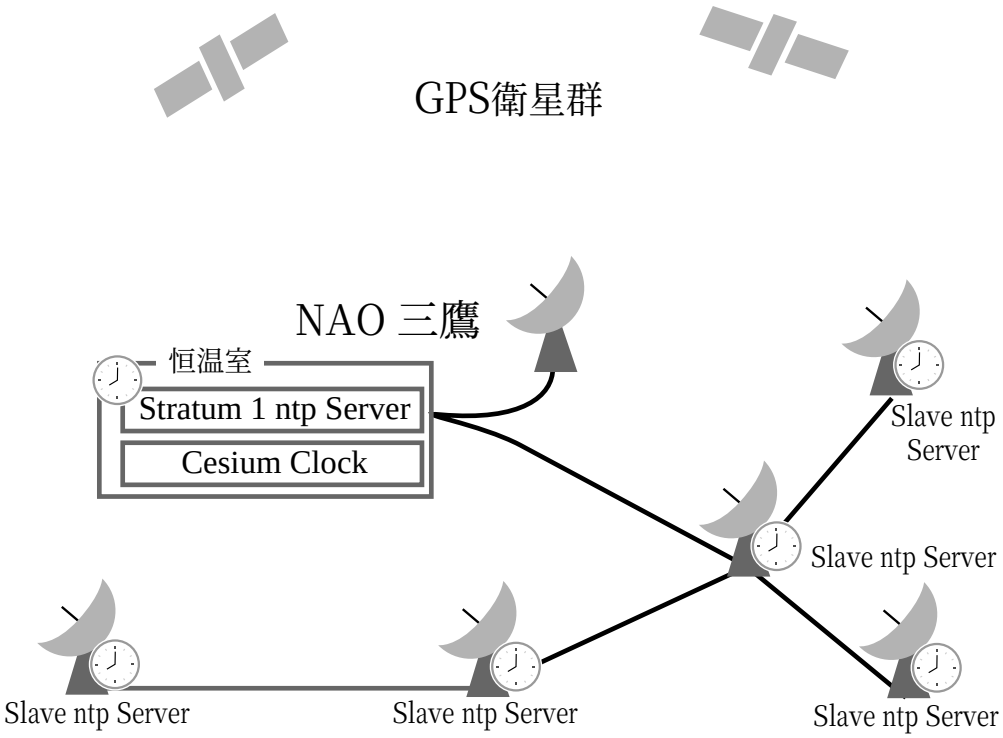

図 7.1: 専用 GPS 受信機に基づく ntp サーバ

注：図ではアンテナを示すのにパラボラアンテナの絵を用いているが、GPS 衛星は静止衛星ではなく複数参照することになるので、アンテナは無指向性である。

第 8 章

おわりに

本年度あらたに創設された NTP ワーキンググループでは、国立天文台の協力を得てセシウム原子時計を時刻情報源とする stratum 1 サーバを試作し、これが WIDE インターネット上から利用可能であることを確認したが、残念ながら今年度は試験運用の域をでることなく終了した。

来年度の最初の目標は、前章でも述べたように定常運用に移行することである。その際、パソコンとワークステーションが連携して時刻情報を提供する現行の暫定的システムを廃し、仕様を全て満たした ntp サーバを DOS マシン上に構築するか、カウンタ回路の出力を直接取り込める UNIX マシンを利用する方針で望みたい。このうち後者を採用する場合、ワークステーションのかわりに BSD/386 あるいは 386BSD が稼働する IBM PC 互換機を利用する予定である。これは以下の理由で「セシウム原子時計から時刻情報を得て稼働する UNIX マシン」として適切であると考えたからである。

- ハードウェアが回路図のレベルで公開されている。
- OS のソースコードが提供されている。
- OS がマイクロ秒の精度で時刻情報を保持できることがはっきりしている[1] 。

このためには、以下の作業が必要になるが、これらはいずれも来年度の早い段階で実現可能である。

- カウンタ回路の出力を IBM PC 互換機に取り込むための高速パラレル入力ボードの調達。
- カウンタ回路の出力をユーザプログラムが参照するための機構の提供。(たとえば、/dev/cesium といったキャラクタデバイスの作成)
- カーネルのソースコードの変更。

また、本年度は検討だけに留まった GPS 受信器を利用した stratum 1 サーバも実装し、WIDE インターネットの各組織が独自に stratum 1 サーバを起動できる環境を構築したい。

[1] 付録 A 参照

謝辞

本研究に対し、平成 4 年度国立天文台留置金の援助をいただきましたことをここに感謝します。

第 9 章

付録: UNIX ワークステーションの内部時計の分解能調査

ntp サーバーの試作の過程で、現行の UNIX ワークステーションの内部時計[1]がどの程度の分解能 (粒度) を持っているかを知る必要が生じたので、これを調査した。

gettimeofday システムコールは、呼び出された時点の内部時刻を struct timeval 型の構造体で返す。そこで、連続して gettimeofday を呼び出し、返される内部時刻の表を作成してみた。得られる値は内部時計の基本単位 — 粒度 — の整数倍であるから、値の変化から粒度を推定することができる。

粒度が非常に粗いか計算機が十分速い場合、すなわち 1 回の呼び出しに要する実時間が粒度より短い場合、内部時刻は一定回数同じ値を示したのち急に粒度の大きさだけ変化する。したがって、内部時刻の増分もしくは第 1 階差 (1 回前の時刻との差) をとると、0 または粒度の整数倍であり、多くの場合粒度そのものとなる。つまり、内部時刻について十分多くの連続したサンプルをとれば、そのサンプルの第 1 階差の 0 でない最小値が粒度を与えることになる。

反対に粒度が荒いか計算機が遅い場合、内部時刻はほぼ一定の時間だけ増えつづけるが増分もまた粒度の整数倍であることから粒度を推定することができる。つまり、上記の方法で第 1 階差の代わりに第 1 階差の増分、すなわち第 2 階差、を用いればよい。ところで、第 1 階差が 0 もしくは粒度の整数倍の場合、第 2 階差の絶対値も 0 もしくは粒度の整数倍になるため、第 2 階差そのままでなく第 2 階差の絶対値を用いることにすれば、粒度と計算機の速度の関係よらず有効である。

このアルゴリズムで動く簡単な C プログラムを以下に示す。

```
#include <stdio.h>
#include <sys/time.h>
#define MAX 1000

void main()
```

[1] 具体的には BSD 系 UNIX の gettimeofday システムコールによって返される値

```
{
        int i, d2, min_d2;
        unsigned int usec[MAX];
        struct timeval tv;
        struct timezone tz;

        for(i = 0; i < MAX ; i++) {
                gettimeof day(&tv, &tz);
                usec[i] = tv.tv_usec;
        }
        min_d 2 = 99999;
        for(i = 2; i < MAX; i++) {
                d2 = usec[ i] - 2*usec[i-1] + usec[i-2];
                if(d2 < 0) d2 = - d2;
                if(d2 > 0 && (i == 2 || d2 < min_d2) ) min_d2 = d2;
        }
        printf("Time Resolution = %d (micro sec)\n", min_d2);
}
```

このプログラムによって国立天文台三鷹[2]及び東京大学教養学部[3] にある各種のワークステーションで得られた結果を表 9.1 に示す。

HP および Sun の製品の場合、粒度は 1 マイクロ秒であり UTC に時計面が合うかどうかを別にするとミリ秒以下の高精度の ntp サーバーとしての資格は十分であると考えられる。DEC 製品の場合は、976 が 1000000/1024 に、また 3906 が 1000000/256 にそれぞれ近いことからビットシフトしたときほぼ 1 秒になるような粒度を採用していることがわかる。

SONY 製品の場合はやや不思議な振舞いを示す。単純にこのプログラムを適用すると NWS821 と PWS1550 のいずれについても 10000 より数十少ない値となり、さらにその値が毎回微妙に変動する。詳しく途中経過を検討すると、ほとんどの場合 tv_usec 値は呼び出されるごとに 1 ずつ増えるため、一見粒度が 1 マイクロ秒のような印象を与える。しかし数十回連続して tv_usec 値が 1 ずつ増えていった後、急に 10000 近く増えて 10000 の整数倍の値にリセットされる。このことから、実際の粒度は 10 ミリ秒と推測できる。 tv_usec 値が呼ばれる度に 1 ずつ増える現象は、おそらく時間計測などの用途で gettimeofday を呼び出すときに差が 0 にならないようにするため同じ tv_usec 値を帰さないようにしているのためと思われる。

[2] mtk.nao.ac.jp ドメイン
[3] c.u-tokyo.ac.jp ドメイン

表 9.1: ワークステーションの内部時計の粒度

機種名	OS+Version	粒度
DEC pc486DX2	BSDI BSD/386 V1.0	1
HP 9000/715	HP-UX 9.01	1
Sun SS-10	Solaris 1.1	1
Sun 4/390	Solaris 1.1	1
Sun Sparc IPX	Sun OS 4.1.2	1
Sun Sparc 1+	Sun OS 4.0.3	1
DEC 3000/400	DEC OSF/1 V1.2	976
DEC 3100	Ultrix 4.2	3906
IRIS Indigo	IRIX 4.0.2	10000
Solbourne S4000	OS/MP 4.1A	10000
Sony NWS3400	NEWS-OS 4.1.2R	10000
Sony PWS1550	NEWS-OS 4.2C	10000
Sony NWS821	NEWS-OS 3.3C	10000
Sony NWS730	NEWS-OS 3.3C	10000
Sun 3/50	Sun OS 4.0.3	10000
Sun 3/80	Sun OS 4.1.1	10000
TOSHIBA SparcLt	OS/AS 4.11	10000
SUMI Station	SEIUX 3.2	− − −

注: 粒度はマイクロ秒の単位で表している。粒度の欄で、− − −はシステムコール等の不備により測定不能であることを意味する。