グループウェアと
ネットワークサービス 46－6
（２００３．１．１６）

時間軸を考慮した知的情報配信システムのモデル化と実装

宮下一博† 後藤文太朗‡
北見工業大学 情報システム工学科
miyakz@pattern.cs.kitami-it.ac.jp† goto@cs.kitami-it.ac.jp‡

我々は、情報配信システムとしての電子メールにおける、「時間」を知的に取り扱うことが出来るシステムの構築を試みた。まず、現実の電子メールの利用に関する記述を行うための世界として、E-Mail World を一階述語論理を用いて記述した。これに時間の概念を導入することで、メールの送信可能時区間、受信可能時区間、配送遅延時間が扱えるようになった。本稿では、メールの送受信時に制約として現れるこれらの時間について考慮した配信方法を示す。また、Temporal Mail、Temporal Mail Server、Temporal Mail Client からなる、回覧メールを実現するプロトタイプシステムについても述べる。

Modeling of an intelligent information delivery system with temporal information and its prototype system

Kazuhiro MIYASHITA† , Fumitaro GOTO‡
Department of Computer Sciences, Kitami Institute of Technology
miyakz@pattern.cs.kitami-it.ac.jp† goto@cs.kitami-it.ac.jp‡

In this paper, we propose an intelligent information delivery system using E-Mail that can deal with temporal information. We describe a simple world about actual activity using E-Mail, called “E-Mail World”, in first order logic. And then we extend system so that we can deal with temporal information. As temporal information, we introduce allowed sending and acceptable and delay time in delivery. This temporal information constrains when send or receive E-Mail. Description of circular E-Mail having time restriction is shown, and delivery planning method of it is given. We also show a prototype system composed from three subcomponents called Temporal Mail, Temporal Mail Server, and Temporal Mail Client.

1. はじめに

今日、電子メールが情報配信メディアとして盛んに利用されている。例えば、電子メール上のアプリケーションとして、メーリングリストや回覧メールなどが組織内の連絡に活用されていることが多く見受けられる。また、メールによるニュースレターに対する反応は Web サイトよりも大きいと言う調査報告[3]もある。

一方、日常生活において我々は、多くの場面で「時間」とかかわりを持っている。例えば、この書類はいつまでに出さなくてはいけないと言った場面がある。広告に関しては、本来の目的である商売等の目的達成の過程において、時間的要因が存在する場合がある。

そこで、我々は情報配信システムとしての電子メールにおいて、「時間」の取り扱いをシステムに取り込むことを試みた。「時間」の取り扱いを組み入れることで、各ユーザの時間にあわせた情報配信という知的な配信が期待できることから、我々はそのシステムを「時間軸を考慮した知的情報配信システム」と呼ぶ。本稿では、現実世界における E-Mail の利用に対し、一階述語論理を用いたモデリングを行う。これにより、時間軸を考慮した電子メール配信のための計画導出が可能になった。さらに、このモデルを回覧メールへ適用し、プロトタイプシステムを設計した。

連絡先 ： 北見工業大学情報システム工学科 , 〒090-8507 北海道北見市公園町 165 番地 Tel: 0157-26-9107 , miyakz@pattern.cs.kitami-it.ac.jp

2. E-Mail World

2.1. 準備

電子メールの利用に関する記述を行うための世界として、E-Mail World を一階述語論理で記述する。この先、個体定数は英大文字または数字から始まる英数字列とする。関数定数は、関数記号または英大文字から始まる英数字列とする。関係定数は数学的演算子または英大文字から始まる英数字列とする。変数は英小文字から始まる英数字列とする。

2.2. Node と Packet と Stream

情報配信の端点を Node オブジェクトと呼ぶ。そして O が Node であることを Node(O)と書く。Node 間を行き来する情報を Packet オブジェクトと呼ぶ。送信元 Node が y で、宛て先 Node が z の Packet が x であることを Packet(x,y,z)と書く。そして、Packet の通り道であるオブジェクトをストリームと呼び、Node y か

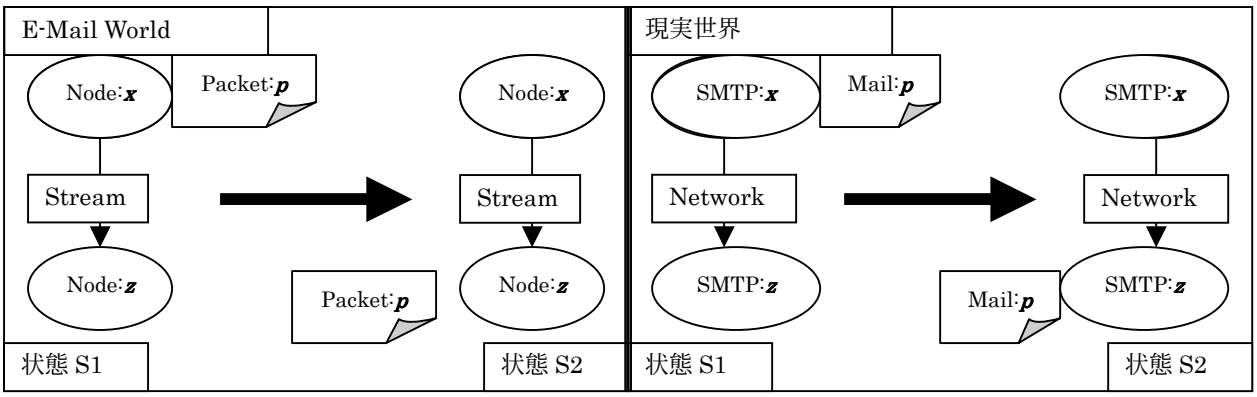

図 2.2-3：E-Mail World 上における情報配信

表：2.2-1：E-Mail World と現実世界の対応

現実世界	E-Mail World
メール	Packet
SMTP サーバ	Node
ユーザ	Node
E-Mail アドレス	属性 Addr
ネットワーク	Stream
メールの保存	属性 Has

ら Node z へのストリームが x であることを Stream(x,y,z)と書く。また、オブジェクトの属性に関する記述のために、関係定数 Attr を用意する。o が、属性値 v の属性 n を持つことを Attr(o , n , v)と書く。Packet の所有関係に関して、Node n が Packet p を持つことを、Attr(n,Has,p)と書く。Node n のアドレスが a であることを、Attr(n,Addr,a)と記す。E-Mail World の状態の性質を表すために、関係 T を導入する。ある項 r が状態 s で真であることを T(r,s)と書く。

ここで、E-Mail World と実世界の対比を表 2.2-1 に示す。ここでは簡略化のため 1 サーバ 1 ユーザ（メールボックス）とし、DNS サーバや IP アドレスなどは考えないものとする。

次に、E-Mail World における E-Mail の送信を記述する。操作子はオブジェクトから行動への関数であり、一群のオブジェクトを、それらを操作する行動へと写像する。また、行動の効果を写像 $Do : A \times S \to S$ の形で概念化できる。(A は行動の全体集合で、S は状態の全体集合）ここで、Node y から Node z に Packet x を移動する Move 操作子を Move(x,y,z)と書くと、その効果は図 2.2-1 のように表すことが出来る。そのフレーム公理は図 2.2-2 になる。Move 操作子による Packet の移動と現実世界におけるメールの配信の対応を図 2.2-3 に示す。

以上の現実世界に対する記述を E-Mail World と呼ぶ。

$$
\begin{aligned}
&T(Node(y),s) \wedge T(Node(z),s) \\
&\wedge T(Attr(y,Has,x),s) \quad \wedge T(Packet(x,y,z),s) \\
&\wedge T(Stream(rt,y,z),s) \\
&\Rightarrow T(Attr(z,Has,x),Do(Move(x,y,z),s))
\end{aligned}
$$

図 2.2-1：Move 操作子の効果

$$
\begin{aligned}
&T(Node(x),s) \Rightarrow T(Node(x),Do(Move(p,y,z),s)) \\
&T(Stream(rt,a,b),s) \Rightarrow T(Stream(rt,a,b),Do(Move(p,y,z),s)) \\
&T(Packet(x,from,to),s) \Rightarrow T(Packet(x,from,to),Do(Move(p,y,z),s)) \\
&T(Attr(x,Has,z),s) \wedge z \neq p \Rightarrow T(Attr(x,Has,z),Do(Move(p,x,y),s)) \\
&T(Attr(o,l,a),s) \Rightarrow T(Attr(o,l,a),Do(Move(p,x,y),s))
\end{aligned}
$$

図 2.2-2：Move のフレーム公理

3. 時間に関する記述

次に、時間に関する概念を形式的に記述する。時間の基本的な要素の集合 T を導入する。文献[1]では、これを Time Domain と呼び、$(T,\leq)$ と記述している。$(T,\leq)$ は instant の集合であり、≤は要素間に成り立つ全順序関係である。本稿では $(Z,\leq)$ とした Time Domain を用いる。次に Time Domain 上の区間について以下のような定義を行う。

定義：時区間

$stp \in Z$, $etp \in Z$, $stp \leq etp$ のとき、$Ti(stp,etp)$ を時区間と呼ぶ。また、時区間の全体集合を TI と表す。■

また、時区間間の関係について、Allen の 13 の関係[2]がある。これを図 3-1 に示す。なお、Stp(m1)は時区間 m1 の stp 値、Etp(m1)は etp 値を返す関数である。また、$m_1, m_2 \in TI$ の時、以下の述語を定義する。

定義：Has_intersection

$$
\begin{aligned}
&Has\_intersection(m_1,m_2) \Rightarrow \\
&\vee During(m_1,m_2) \vee Contains(m_1,m_2) \vee Overlapps(m_1,m_2) \\
&\vee Overapped\_by(m_1,m_2) \vee Meets(m_1,m_2) \vee Met\_by(m_1,m_2) \\
&\vee Starts(m_1,m_2) \vee Started\_by(m_1,m_2) \vee Finishes(m_1,m_2) \\
&\vee Finished\_by(m_1,m_2) \vee Equal(m_1,m_2)
\end{aligned}
$$

■

以降、本稿において「時間」や「時刻」と言った場合、Time Domain 上のある $t \in Z$ のことを言い、時

表 4.1-1：各オブジェクトと関連する時間属性例

タイプ	時間属性	意味
Node	SendableTime	Packet 送信可能時区間
	AcceptTime	Packet 受信可能時区間
	WellReading-Time	Packet を良く読む時区間
Packet	DesiredTime	宛先ユーザに読んで欲しい時間
	SentTime	Node から送られた時区間
	ArrivalTime	Node に到着した時区間
Stream	DelayTime	遅延時間

1 ： Equal(m1 , m2) iff
　　Stp(m1) = Stp(m2) and Etp(m1) = Etp(m2)
2 ： Before(m1 , m2) iff Etp(m1) < Stp(m2)
3 ： After(m1 , m2) iff Before(m2 , m1)
4 ： During(m1 , m2) iff
　　Stp(m2) ≦ Stp(m1) and Etp(m1)≦Etp(m2)
5 ： Contains(m1 , m2) iff During(m2 , m1)
6 ： Overlaps(m1 , m2) iff
　　Stp(m1)≦Stp(m2)≦Etp(m1) and
　　Stp(m2)≦Etp(m1)≦ Etp(m2)
7 ： Overlapped_by(m1, m2)iff Overlaps(m2 ,m1)
8 ： Meets(m1, m2) iff Etp(m1) = Stp(m2)
9 ： Met_by(m1 , m2) iff Meets(m2 , m1)
10： Starts(m1 , m2) iff Stp(m1) = Stp(m2)
11： Started_by(m1 , m2) iff Starts(m2,m1)
12： Finishes(m1, m2) iff Etp(m1) = Etp(m2)
13： Finished_by(m1,m2) iff Finishes(m2 , m1)

図 3-1：Allen の時区間関係

区間と区別する。また「時間帯」と言った場合、それは時区間の事を指す。

4. E-Mail World の拡張

4.1. E-Mail World への時間の導入

前節で述べた時間の概念を E-Mail World に導入する。各オブジェクトについて時間属性の例を表 4.1-1 に示す。Node に Packet の送信可能時区間と受信可能時区間を持たせ、Stream には遅延時間を取り入れ、Packet には、Node から送られた時間と宛先に到着した時間を持たせることにする。

状態とはある時刻での世界のスナップショットを表す事から、時刻と状態について $TS: Z \to S$ なる関数を

$$
\begin{aligned}
&T(Attr(y,Has,x),TS(t))\\
&\quad \wedge T(Packet(x,y,z),TS(t))\\
&\quad \wedge T(Stream(st,y,z),TS(t)) \wedge T(Attr(st,DelayTime,dt),TS(t))\\
&\quad \wedge T(Attr(y,SendableTime,st),TS(t)) \wedge T(Attr(z,AcceptTime,a),TS(t))\\
&\quad \wedge Has\_intersection(st,Ti(t,t)) \wedge Has\_intersection(a,Ti(t+dt,t+dt))\\
&\Rightarrow\\
&T(Attr(z,Has,x),Do(Move(x,y,z,t),TS(t)))\\
&\quad \wedge T(Attr(x,SentTime,From(y,t)),TS(t))\\
&\quad \wedge T(Attr(x,ArrivalTime,At(z,t+dt)),Do(Move(x,y,z,t),TS(t)))
\end{aligned}
$$

図 4.1-1：時間を考慮した Move の効果

$$
\begin{aligned}
&T(Node(x),TS(t)) \Rightarrow T(Node(x),Do(Move(p,y,z,t),TS(t)))\\
&T(Stream(st,a,b),TS(t)) \Rightarrow\\
&\qquad T(Stream(st,a,b),Do(Move(p,y,z,t),TS(t)))\\
&T(Packet(x,from,to),TS(t)) \Rightarrow\\
&\qquad T(Packet(x,from,to),Do(Move(p,y,z,t),TS(t)))\\
&T(Attr(x,Has,z),TS(t)) \wedge z \neq p \Rightarrow\\
&\qquad T(Attr(x,Has,z),Do(Move(p,x,y,t),TS(t)))\\
&T(Attr(o,l,a),TS(t)) \Rightarrow T(Attr(o,l,a),Do(Move(p,x,y,t),TS(t)))
\end{aligned}
$$

図 4.1-2：時間を考慮した Move のフレーム公理

$$
\begin{aligned}
&Delay(Move(x,y,z,t)) = dt\\
&\qquad if\ T(Node(y),TS(t)) \wedge T(Node(z),TS(t))\\
&\qquad \wedge T(Packet(x,y,z),TS(t))\\
&\qquad \wedge T(Stream(str),TS(t))\\
&\qquad \wedge T(Attr(str,DelayTime,dt),TS(t))
\end{aligned}
$$

図 4.1-3：時間を考慮した Move 行動が引数の時の Delay 関数

用意し、状態を時刻で表現する。

また、行動の完了に必要な時間についても考慮する必要がある。Node 間に遅れ時間が存在することから、Move 行動にも遅れが生じる。遅れを表現するのに、$Delay : A \to Z$ を導入する。これは行動から、その行動の完了までにかかる時間に写す。また、Do 関数の概念を拡張し、行動の遅れを考慮した行動の効果を表すようにする。さらに、行動ブロックの概念も導入する。以上より、Do は以下のようになる。

定義： Do 関数

$$
\begin{aligned}
&Do(a,TS(t)) = TS(Delay(a)+t)\\
&Do([],TS(t)) = TS(t)\\
&Do([a \mid l],TS(t)) = Do(l,Do(a,TS(t)))
\end{aligned}
$$

■

これにより、遅れ時間を考慮した行動の表現が可能になる。たとえば、$a \in A, Delay(a) = 6$ のとき、$Do([a],TS(1))$ は状態 $TS(1)$ から $TS(7)$ に写す。

次に、Move オペレータについても時間を考慮したものに拡張し、引数に送信時刻 t を入れる。その効果を図 4.1-1 に示す。なお、$From(y,t)$ は時刻 t に y から送信されたことを示す項で、$At(z,t)$ は z に時刻 t で到着したことを示す項である。また、フレーム公理も時

間を考慮したものになる。これを図 4.1-2 に示す。また、Delay 関数も図 4.1-3 に示す。

以上の記述により、グリーンの方法[5]を用いた情報配信のためのプラン作成を行う。図 4.1-4 において、
$Goal(a) \Leftrightarrow T(Attr(P, ArrivalTime, At(N_2, x)), Do(a, TS(t)))$
を目標とした計画導出を図 4.1-5 に示す。

4.2. 順序が固定された配送

E-Mail World において、Packet x を所有する Node が、Move 操作子により、Node N_0, N_1, N_2,..., N_k の順に変わっていくことを順序が固定された配送と呼ぶ。Packet オブジェクト x において、この順序が固定された配送の指定を、Packet(x, N_0, [N_1, N_2,.., N_k])と記述する。

順序が固定された配送を考慮した E-Mail World に前節と同様に時間の導入を行う。この際、Move に時間制約があるので、配送リストの要素順に Move すると途中で失敗する可能性がある。そこで、配送リストが以下の述語を真とすれば時間制約を満したものであるとし、配送が可能であると言う。

定義： Select(n, t)

$$
\begin{aligned}
&Select([x], t)\\
&Select([n, m \mid o], t) \Rightarrow\\
&\quad T(Attr(n, SendableTime, x), TS(t))\\
&\wedge T(Attr(m, AcceptTime, y), TS(t))\\
&\wedge (Overlaps(x, y) \vee Before(x, y)) \wedge Select([m \mid o], p, t)
\end{aligned}
$$
■

4.3. 順序が固定されない配送

E-Mail World において、Packet x を所有する Node が、Move 操作子により、Node N_0 を起点として，N_1, N_2,...,N_k の Node のどれか一つずつ変わっていくことを順序が固定されない配送と呼ぶ。この順序が固定されない配送の指定を、Packet(x, N_0, < N_1, N_2,.., N_k >)と記述する。

順序が固定されない配送を考慮した E-Mail World に前節と同様に時間の導入を行う。この際、配送リストの要素順を変えてよいならば、元のものから時間制約を満した別のリストを生成する方法を考えることができる。

Perm(n,m)はリスト n の順列のリストが m であることを表す。また。以下の述語を定義する。

定義： Dest(i,o,t)

$$
\begin{aligned}
&Dest(i, o, t) \Rightarrow Perm(i, p) \wedge Dest2(p, o, t)\\
&Dest2(<>, <>, t)\\
&Dest2(< a \mid o >, < a \mid p >, t) \Rightarrow Select(a, t) \wedge Dest2(o, p, t)\\
&Dest2(< a \mid o >, p, t) \Rightarrow Dest2(o, p, t)
\end{aligned}
$$
■

また、SendableTime や AcceptTime は、現実世界の情報配信においては仮定の値であり、Node に相当す

$$
\begin{aligned}
&T(Node(N_1), TS(1)), T(Node(N_2), TS(1))\\
&T(Packet(P, N_1, N_2), TS(1)), T(Attr(N_1, Has, P), TS(1)),\\
&T(Stream(STR, N_1, N_2), TS(1)),\ Has\_intersection(Ti(5,10),\ Ti(5,5)),\\
&T(Attr(N_1, SendableTime, Ti(0,3)), TS(1)),\\
&T(Attr(N_1, AcceptTime, Ti(2,5)), TS(1)),\\
&T(Attr(N_2, SendableTime, Ti(0,3)), TS(1)),\\
&T(Attr(N_2, AcceptTime, Ti(5,10)), TS(1)),\\
&T(Attr(STR, DelayTime, 4), TS(1))
\end{aligned}
$$

図 4.1-4: E-Mail World の例

$$
\begin{aligned}
&1: \{\neg Goal(a), Ans(a)\}\\
&2: \{\neg T(Attr(P, ArrivalTime, At(N_2, x)), Do(a, TS(t))), Ans(a)\}\\
&3: \{\neg T(Attr(y, Has, p), TS(t)), \neg T(\neg Attr(z, Has, p), TS(t)),\\
&\quad \neg T(Packet(p, y, z), TS(t)), \neg T(Stream(str, y, z), TS(t)),\\
&\quad \neg T(Attr(str, DelayTime, d), TS(t)),\ \neg T(Attr(y, SendableTime, st), TS(t)),\\
&\quad \neg T(Attr(z, AcceptTime, a), TS(t)), \neg(\ Has\_intersection(st\ , Ti(t,t))),\\
&\quad \neg(\ Has\_intersection(a\ , Ti(t+d, t+d))),\\
&\quad T(Attr(p, ArrivalTime, At(z, t+d)), Do(Move(p, y, z, t), TS(t)))\\
&\quad Ans(Move(p, y, z, t))\}\\
&4: \{T(Attr(N_1, Has, P), TS(1))\}\ \ 5: \{T(\neg Attr(N_2, Has, P), TS(1))\}\\
&6: \{T(Packet(P, N_1, N_2), TS(1))\} 6: \{T(Stream(STR, N_1, N_2), TS(1))\}\\
&7: \{T(Attr(STR, DelayTime, 4), TS(1))\} 8: \{T(Attr(N_1, SendableTime, Ti(0,3)), TS(1))\}\\
&9: \{T(Attr(N_2, AcceptTime, Ti(2,5)), TS(1))\}\\
&10: \{\ Has\_intersection(Ti(5,10),\ Ti(5,5))\}\\
&11: \{Ans(Move(P, N_1, N_2, 1))\}
\end{aligned}
$$

図 4.1-5：計画の導出

$$
\begin{aligned}
&T(Attr(y, Has, x), TS(t))\\
&\wedge T(Packet(x, y, < z, v \mid u >), TS(t))\\
&\wedge T(Stream(st, y, z), TS(t)) \wedge T(Attr(st, DelayTime, dt), TS(t))\\
&\wedge T(Attr(y, SendableTime, st), TS(t)) \wedge T(Attr(z, AcceptTime, a), TS(t))\\
&\wedge Has\_intersection(\ st, Ti(t,t)) \wedge Has\_intersection(a\ , Ti(t+dt, t+dt))\\
&\Rightarrow\\
&T(Attr(z, Has, x), Do(Move(x, y, z, t), TS(t)))\\
&\wedge T(Attr(x, SentTime, From(y, t)), TS(t))\\
&\wedge T(Attr(x, ArrivalTime, At(z, d+t)), Do(Move(x, y, z, t), TS(t)))\\
&\wedge T(Packet(x, z, < v \mid u >), Do(Move(x, y, z, t), TS(t)))\\
&\wedge Move(x, z, v, Delay(Move(x, y, z, t)) + t)\\
&\\
&\\
&T(Attr(y, Has, x), TS(t))\\
&\wedge T(Packet(x, y, < z \mid u >), TS(t))\\
&\wedge T(Stream(st, y, z), TS(t)) \wedge T(Attr(st, DelayTime, dt), TS(t))\\
&\wedge T(Attr(y, SendableTime, st), TS(t)) \wedge T(Attr(z, AcceptTime, a), TS(t))\\
&\wedge (\neg Has\_intersection(\ st, Ti(t,t)) \vee \neg Has\_intersection(a\ , Ti(t+dt, t+dt)))\\
&\Rightarrow\\
&Dest(< y, z \mid u >, << m, n \mid l >| k >, t)\\
&\wedge T(Packet(x, m, < n \mid l >), TS(t))\\
&\wedge Move(x, m, n, t)
\end{aligned}
$$

図 4.3-1：宛先リストの更新を考慮した Move の効果

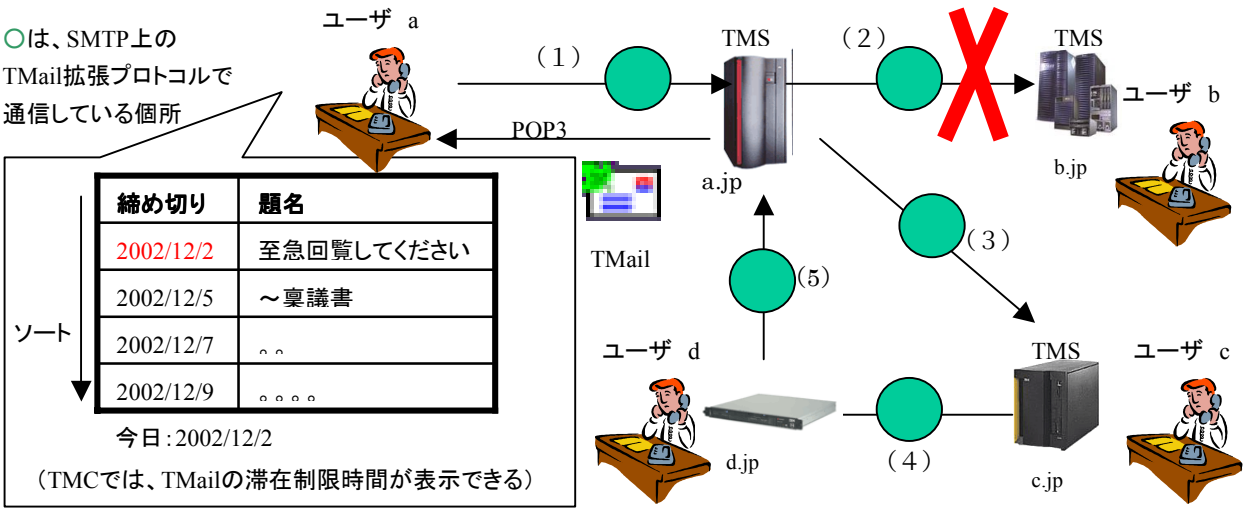

図 5-1：システム概念図

表：5-1：システムコンポーネントと E-Mail World の対応

コンポーネント	機能	E-Mail World
TMail	回覧メールを表す	Packet
TMS	配信プランを作成して、配信する	Node
TMC	メールクライアント。滞在制限時区間を表示（締め切り）	なし
ネットワーク	メールの転送	Stream

る装置の故障などにより、配信が失敗することが考えられる。このような場合に対処するため、Move 行動が失敗した時に Dest で宛先リストを作り直し、送信を再開できるように、Move 操作子を拡張する。これを図 4.3-1 に示す。

4.4. 回覧メールの表現

前節の記述を基に、現実世界の回覧メールを表現する。回覧メールの特徴について以下に列挙する。

一般的な特徴

1. 宛先に順番がある。

時間的観点から見た特徴

2. メールに一巡制限時区間がある。
3. 一巡制限時区間から、メールが各人に滞在することの出来る滞在制限時間がある。

回覧の方法

1. 順番厳守
2. 時間厳守

一巡制限時区間を守ることができる。ただし、次の順番の受け取り許可時間にメールを送られない場合は、順番をスキップする必要がある。つまり、メールのあて先がダイナミックに変化する。

宛先のダイナミクスは、Move で表現する。また、

```
From: a@a.jp
X-SeqTo: b@b.jp,c@c.jp, a@a.jp
X-CycleLimitTime: 1,10
X-PlanSeq:Move(b@b.jp,1),Move(c@c.jp,4),Move(a@a.jp
        ,8)
X-TMail-Version: 0.1
X-TMail-ID: 1

Hello , This is a TMail Example !! !
```

図 5-2：TMail の例

一巡惰性現時区間は Packet に CycleLimitTime 属性、滞在制限時間は StayLimitTime 属性を持たせる。

5. プロトタイプシステムの設計

回覧メールシステムの設計を行う。システムは大きく分けて、Temporal Mail（TMail）と、Temporal Mail Server（TMS）、そして Temporal Mail Client（TMC）で構成される。図 5-1 にシステムの概念図を示す。

なお、システムにおける時刻は正の 64 ビット整数で表す。仮定として、各 TMS 間の時差は無いものとし、全 Node の SendableTime と AcceptTime は Ti(0,MAX) とする。MAX は 64 ビット整数の正の最大値である。

また、表 5-1 に各コンポーネントと E-Mail World のオブジェクトの対応を示す。ここで、TMC を Node と見なす考え方もあるが対応無しとしている。これは、TMC は能動的にメールを受信するのに対し、Node は受動的に Packet を受信する性質から、両者の一致は不自然と考えたためである。

次に TMail の例を図 5-2 に示す。TMail の属性は拡張ヘッダフィールドとして保持している。配信プランの作成は、TMail に含まれる宛先リスト(X-SeqTo)の情報を元にして行う。例えば Move(a@a.jp,7)は時刻 7 に a@ajp にメールを配信するプランを表す。また、

表 5-2：拡張コマンド一覧

コマンド	パラメタ	機能
TMAIL	なし	送信するメールがTMailである
TDELE	TMail-ID	回覧停止、TMail 削除

X-CycleLimitTime は一巡制限時区間である。

プラン生成について、システムの簡略化のため、今回は推論によるプランニングを行わずに、次のような簡便な方法を用いた。宛先リストが$[a_1,...,a_n]$で、一巡制限時区間が $Ti(s,l)$。そして、プラン生成時刻が t の時、各アドレスへの送信時間は次の関数で求められる。

$$f(k)=\left[\, t+\frac{|l-t|}{n}(k-1)\,\right] \quad \cdots\cdots\cdots\cdots\cdots\cdots(1)$$

ただし、n はリストの長さであり、k は要素の添え字 $(1\le k\le n)$ である。例えば、宛先リストが[b@b.jp,c@c.jp, a@c.jp]で一巡制限時区間が $Ti(1,10)$、プラン生成時刻が 1 のとき、プランは次のようになる。[Move(b@b.jp,1),Move(c@c.jp,4),Move(a@a.jp,8)]。なお、各アドレスの送信時間が各 TMS における TMail の滞在制限時間となる。

図 5-3 に TMS のアーキテクチャを示す。これは Java 言語を用いて実装されており、James[4]を利用している。回覧メールの配信部分は TMS-Mailet で行う。ここでは、TMail ヘッダの解釈や宛先リストの更新、それに伴うプランの生成などを行う。また、拡張プロトコルの部分に関しては、独自の SMTP Handler を用意している。

次に、配信処理について大まかに説明する。図 5-1 において、最初にユーザ a は TMail を送信する。新規 TMail に対して a.jp はプランを生成する。この時の TMail が図 5-2 であるとする。そして、プランに従い b.jp に送信する。しかし通信不能であるので、プランを再生成し、a@a.jp の前に d@b.jp に送信するように X-SeqTo が更新される。後に、c@c.jp、d@c.jp ,a@a.jp の順に送信される。

一巡制限時区間内で配送を完了する機能については、ユーザがメールを次の人物に送信しなくても、滞在制限時間を過ぎると TMS が自動的に次へ送信する機能により実現する。例えば図 5-2 の場合、ユーザ c が時刻 4 までに次の宛先に送信しないような場合、TMS が自動的に送信する。

また、図 5-1 の○部分はシステム独自の SMTP 拡張プロトコルで通信が行われる。表 5-2 に拡張コマンド一覧を示す。

6．おわりに

現実世界における E-Mail の利用を、一階述語論理で記述し、時間の概念を導入した。時間軸を考慮した電子メールによる配信のための計画を導出できることを

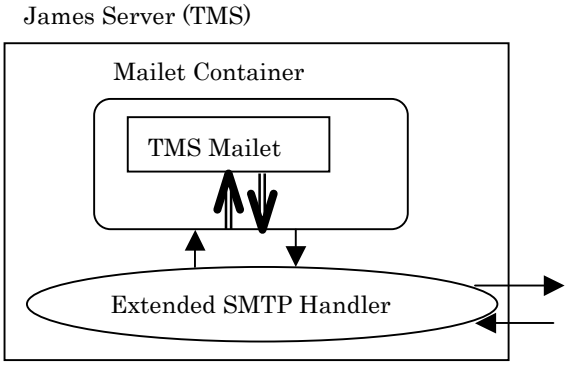

図 5-3：TMS アーキテクチャ

示し、プロトタイプシステムを設計した。システムの特徴は、一巡制限時区間内でメールを回覧できることである。

本稿で示した E-Mail World の設定では、一巡制限時間内で配送が不可能になる場合を想定していなかった。すなわち、宛先サーバがすべて通信不能で、通信可能な代替が無い時である。このような状況に対する対処は今のところシステムでは考慮されていない。また時差や多対多の配信が考慮されていない。

今後は、システムの実装と利便性に関する実験を行い、その評価を行う予定である。さらに、E-Mail World の拡張を行い、システムの機能強化を図っていく予定である。

参考文献

[1] Claudio Bettini, Sushil Jajodia, Xiaoyang Sean Wang：”Time Granularities in Databases, Data Mining, and Temporal Reasoning”, Springer-Verlag New York Inc.,2000

[2] Allen,J.: Maintaing Knowledge about Temporal Intervals, Communications of theACM,Vol.26,No.11,pp.832-843

[3] Nielsen Norman Group:”E-Mail Newsletter Usability 79 design guidelines for subscription, newsletter content and account maintenance based on usability studies” http://www.nngroup.com/reports/newsletters/

[4] Apache Jakrta Project: “The Java Apache Mail Enterprise Server (a.k.a. Apache James)” http://jakarta.apache.org/james/

[5] Green,C.,: “Theorem-Proving by Resolution as a Basis for Question-Answering Systems,” in Meltzer , B. and Michie , D.(eds.),Machine Intelligence 4. Edinburgh, UK: Edinburgh University Press, 1969,pp.183-205