ペトリネットによる知能リンクの動作解析

A Behavior Analysis of an Intelligent Link by Using Petri Nets

宮平知博　中村維男　重井芳治
Tomohiro MIYAHIRA　Tadao NAKAMURA　Yoshiharu SHIGEI
東北大学　工学部
Faculty of Engineering, Tohoku University

1. はじめに

最近，“通信”の概念と“処理”の概念を融合した機能を有する知能リンクが提案されている[1][2]。この知能リンクはソフトウェアによる機能供給型汎用パイプライン処理システムである。一方，ペトリネットはその図形表現による視覚的理解の容易さ，解析手段の明確化，階層的記述の可能性等によって，近年になって非同期システムを表現する優れた方法として注目を浴びている[3]。本稿では，タイムペトリネットを用いたシミュレーションによって，スループット，滞在時間等の評価基準を中心とした知能リンクの動作解析を行なう。

2. 知能リンクとペトリネット

2．1　知能リンク

知能リンクは，ホスト計算機A，B間に，伝送路とデータの処理を行なうプロセッサからなるプロセス要素を1次元にn段並べて構成される。任意のデータはパイプライン処理されながら伝送されるが，その際，各データの処理はデータに付随して供給されるプログラムによって行なわれる。従って，知能リンクはソフトウェアによる機能供給型の汎用パイプラインシステムである。

このように，知能リンクでは各セグメントの処理をプログラムによって与えるため，各セグメントで処理時間が一定とならず，非同期型のパイプラインとして動作することになる。そのために，図1に示すようにセグメント間にブロッキングが生じ，システムの性能が低下することが予想される。

このブロッキングを避けるための方法として，セグメント間にバッファを設けてセグメントによる処理時間のばらつきを吸収させることが考えられる。それにより，スループットが向上することが見込まれる。

これを次章以降でペトリネットを用いて解析する。

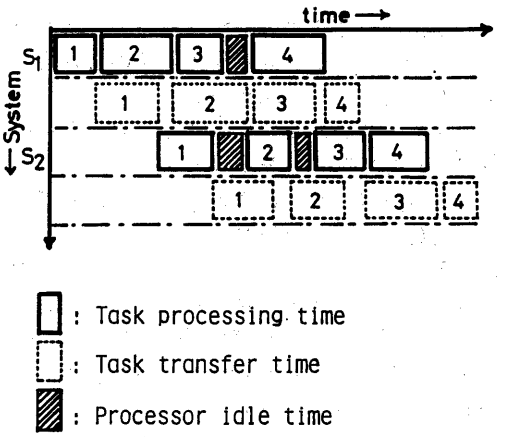

図1．知能リンクのタスク遷移

２．２　ペトリネット

ペトリネットは、1962年にPetriによって用いられて以来，情報の流れの抽象的形式的モデルとして知られている。近年になって、非同期なシステムを表現する優れた方法として注目され，各方面への応用が展開されている。

ペトリネットπは，次の３つの集合の組である。

π＝（Ｐ，Ｔ，Ａ）

ただし

Ｐ：有限個の場所の集合。

Ｔ：有限個の遷移の集合。

Ａ：有限個の有向枝の集合。各枝は遷移から場所へ，あるいは，場所から遷移へ向かう。

図示する時には、場所は○（円）で、遷移は｜（棒）で、有向枝を→（矢印）で表わす。さらに、場所にトークン（図示する場合には・（点）で表わす）と呼ばれる目印を与えることで、ペトリネットで表わされたシステムの状態を表わすことができる。場所にトークンを与えることをマーキングと呼び、マーキングされたペトリネットをマーク付きペトリネットと呼ぶ。

ペトリネットは遷移を発火することによって実行される。遷移は，そのすべての入力場所が少なくとも１つのトークンを持つ時に発火可能となる。遷移が発火すると，その入力場所から１つずつトークンを取り除き，その出力場所にトークンを１つずつ加える。

マーキングmのもとである発火可能な遷移を発火することでマーキングm'が得られる時，m'はmから直接到達可能であるという。m'がmから直接到達可能であるか，または，mから直接到達可能なマーキングから到達可能である時，m'はmから到達可能であるという。マーキングを表わす節点を直接到達可能性を示す有向枝で結んだ木を到達可能木と呼び，ペトリネットの解析によく用いられる。

また、どの場所にも１個より多いトークンが存在しない時，そのペトリネットは安全であるという。

３．タイムド・ペトリネットとタイムペトリネット

ペトリネットは時間に関する記述を含んでいない。しかし，一般のシステムをモデル化する際には時間の概念の導入が望まれる。ペトリネットに時間に関する記述を導入したものに、タイムド・ペトリネットとタイムペトリネットがある。これら２つのペトリネットに関してこの章で述べる。

３．１　タイムド・ペトリネット

タイムド・ペトリネットはRamchandaniによって提案されたもので、ペトリネットの各遷移に発火時間を割り当てたものである[4],[5]。

タイムド・ペトリネットに於てそれぞれの遷移t_iには発火時間$f(t_i)$が割り当てられている。遷移t_iが発火可能になると，t_iの入力場所のそれぞれからトークンを取り去ることで発火が開始され，このトークンは時間$f(t_i)$の間t_iに留まり，t_iの出力場所の各々にトークンを加えることで発火が終わる。そして，発火のそれぞれはそれが発火可能となると同時に開始される。

タイムド・ペトリネットは各遷移の発火が発火時間関数$f(t_i)$によって一意に決まるので，解析が容易であり，実際のシステムの動きとペトリネットの遷移を関連づけて考え易い。しかしながら，発火中にトークンが遷移中に留まるためにペトリネットの到達可能木との対応がつかない上に，次に示すタイムペトリネットと異なり発火の優先権を表わすことはできない。

３．２　タイムペトリネット

タイムペトリネットはMerlinらによって提案されたもので、ペトリネットの各遷移t_iに発火時刻許容範囲$[\tau_i^*, \tau_i^{**}]$を与えたものである[6]。

遷移t_iは次の条件(1),(2)によって瞬間的に発火する。

(1)時刻τでτ$_i$が通常のペトリネットで発火可能（これを以下マーキング条件が成立したという）である。

(2)t_iのマーキング条件が成立した時刻をτとすると，$\tau+\tau_i^* \leq \tau \leq \tau+\tau_i^{**}$ の範囲である。

タイムペトリネットは，発火時刻許容範囲の

違いによって発火の優先権を表わし得る。しかし、元来の利用が、通信制御手順におけるコマンド不達に対するタイムアウト時間の設定問題、検証問題を目標にしていたので、計算機の動作解析には適用しにくい。

3.3 タイムペトリネットの制限

タイムペトリネットの解析を容易にし、次章以降で知能リンクの解析を行なうために、

$$\tau_i^* = \tau_i^{**} = f(t)$$

という制限を加える。つまり、遷移t_iはマーキング条件が$f(t)$時間成立すると、その瞬間に発火する。この制限を加えたタイムペトリネットを以下で単にＴＰＮと呼ぶ。

タイムド・ペトリネットの遷移は、このＴＰＮの２つの遷移と１つの場所によって図２のように書き直すことができる。従って、すべてのタイムド・ペトリネットはＴＰＮに書き直すことができる。

以下では安全なＴＰＮだけを考える。

ＴＰＮの時点表示（ＩＤ）d_iを次のように２つの関数の組として定義する。

$d_i = (m_i, w_i)$

ただし

m_i：場所にトークンの有無を対応づけるマーキング関数

w_i：遷移に発火までに待たなければいけない最少の待ち時間を対応させる待ち時間関数

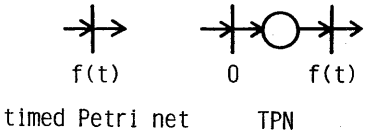

図２．タイムド・ペトリネットとＴＰＮの対応

遷移t_kは発火までに最低$w_i(t_k)$待たなければならないが、$w_i(t_k)$経ったからといって発火できるとは限らず、その間連続してマーキング条件が成り立っている場合にのみ発火する。

以下のことが成り立つ場合に$d_i = (m_i, w_i)$から$d_j = (m_j, w_j)$へ直接到達可能であるという。

(1) $\forall p \in P \quad m_j(p) = m_i(p) + \sum_{t \in In(p)} e_i(t) - \sum_{t \in Out(p)} e_i(t)$

(2) $\forall t \in T \quad w_j(t) = \begin{cases} f(t) & t \in \overline{S(m_i)} \cup \overline{S(m_j)} \cup S(m_i) \\ w_i(t) - \tau_{ij} & \text{その他} \end{cases}$

ただし、

$\tau_{ij} = \min_{t \in S(m_i)} w_i(t)$

$S(m_i) = \{m_i$でマーキング条件が成立している遷移$\}$

$S'(m_i) = \{S(m_i) \ni t, w_i(t) = \tau_{ij}$なるも。ただし衝突のある場合はその内の１つ$\}$

$e_i(t) = \begin{cases} 0 & t \notin S'(m_i) \\ 1 & t \in S'(m_i) \end{cases}$

次章で、このＴＰＮによって知能リンクをモデル化する。

4. 知能リンクのペトリネットモデル

4.1 バッファのない場合

処理要素が２段の場合の知能リンクのペトリネットモデルを図３に示す。各遷移は次のような意味を持つ。

t_1：ホスト計算機の送信

t_2, t_9, t_{16}：転送終了

t_3, t_4, t_{10}, t_{11}：処理開始

t_5, t_6, t_{12}, t_{13}：処理終了

t_7, t_{14}：送信

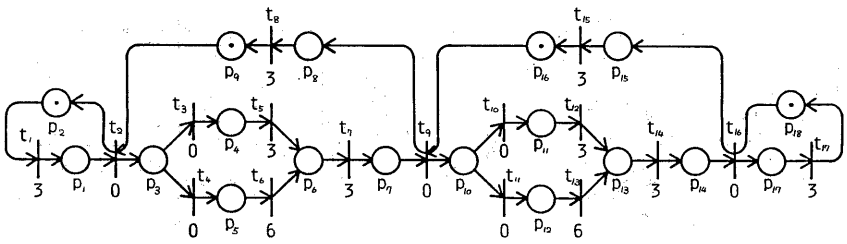

図３．バッファのない場合の知能リンクのペトリネットモデル

t_8, t_{15}　：受信
t_{17}　：ホスト計算機の受信
また、各場所は次のような状態に対応する。
p_1　：ホスト計算機の送信中
p_2　：ホスト計算機の送信準備
p_3, p_{10}　：処理待ち
p_4, p_5, p_{11}, p_{12}：処理中
p_6, p_{13}　：送信準備
p_7, p_{14}　：送信中
p_8, p_{15}　：受信準備
p_9, p_{16}　：受信中
p_{17}　：ホスト計算機の受信準備
p_{18}　：ホスト計算機の受信中

t_2からt_9, t_9からt_{16}までがそれぞれ知能リンクの1段に対応する。各段では受信・処理・送信で1つのサイクルをつくっており、受信，送信はそれぞれ前段の送信、次段の受信とタイミングを取って終了する。処理は長さの違う2種類があるものとし、その内のどちらか一方が行なわれる。また、各遷移の発火待ち時間は図中のそれぞれの遷移のそばに数字で示してある。

段数が2段より多い場合には、t_2からt_9までと全く同様の遷移及び場所が対応する数だけ直列に並ぶ。

4．2　バッファがある場合

各段での処理時間のばらつきを吸収させるために、セグメント間にバッファを設けてその効果を確かめる。1単位のプログラム＋データを格納することができるバッファを受信の前に1つ置いたシステムを考え、図3と同様に処理要素が2段の場合の知能リンクのペトリネットモデルを図4に示す。

図3に対して新たに加わった遷移及び場所は次のような意味を持つ。

t_{18}, t_{19}　：送信終了
p_{19}, p_{21}　：バッファFULL
p_{20}　p_{22}　：バッファEMPTY

バッファが2つ以上の場合には、バッファ1つにつき1段当り2つの場所と1つの遷移が同様にに付け加えられる。バッファリングのための時間はかからないものとしているので、サイクルタイム、滞在時間等はバッファのない場合と単純に比較することができる。

5．解析結果

5．1　2段の場合

2段でバッファのない場合（図3）の到達可能IDのグラフ表現を図5に与える。各IDは単純な数字で代用し、（m，w）の詳細は省略する。本来，各々の枝は発火遷移でラベルづけされれば十分であるが、各IDの内容を省略したので2ノード間の時間もラベルとして各枝に与えてある。

この到達可能IDから1段の平均サイクルタイムを求めると、

$9+3(P(t_{11})+P(t_4)P(t_{10}))$

となる。ただし、$P(t_{11})$, $P(t_{10})$, $P(t_4)$はそれぞれt_{11}，t_{10}，t_4の発生確率であり、

$P(t_{11})+P(t_{10})=P(t_4)+P(t_3)=1$

である。

また、1段目で受信を完了してから2段目で送信を完了するまでの時間、すなわち滞在時間は、

$12+6(P(t_{11})+P(t_4)P(t_{10}))$

となる。

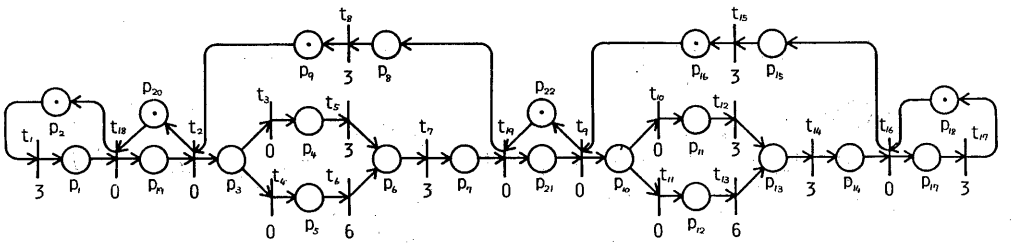

図4．バッファのある場合の知能リンクのペトリネットモデル

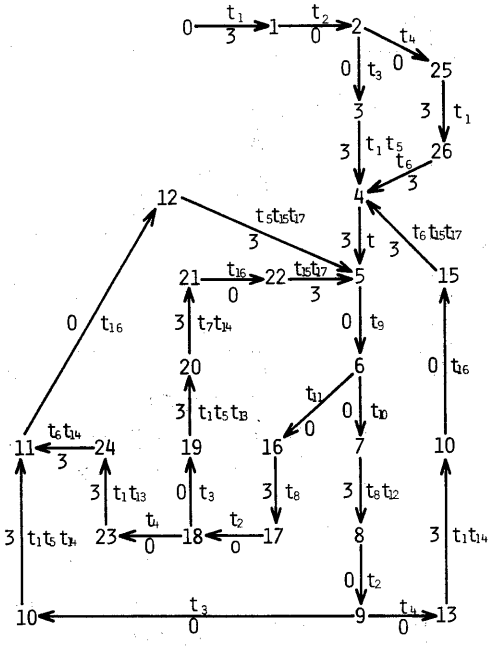

図5. 図3のTPNの到達可能ID

5．2　シミュレーション結果

2段の場合は前節のように到達可能IDのグラフ表現からサイクルタイム、滞在時間などが容易に導き出せるが、ペトリネットが大規模になると解析が簡単ではない。従って、今回の解析はシミュレーションによって行なった。

シミュレーションは、ペトリネットを直接に実行させること、すなわち、発火可能な遷移を次々と発火させることを基本としている。そして、2つの処理のどちらかの選択は乱数によって行なっている。

5．2．1　バッファのない場合

処理時間の短い方の処理の開始、t_3, t_6…の発生確率の変化による平均サイクルタイムの変化を処理要素の段数を変えて示したのが図6である。各段での処理の発生確率は等しいものと仮定しており、短い処理の発生確率が大きくなればサイクルタイムは当然小さくなる。しかし、段数が増えると、次段以降の処理が長いためのブロッキングや前段の処理が長いことによるアイドル状態などのために、サイクルタイムが長くなる。N=1の場合がそのような性能の低下が全くない場合に相当し、それ以上にかかっている時間は、各段の処理要素が遊んでいる時間ということになる。

1段当りの滞在時間をt_3, t_6…の発生確率及び段数を変えて示したのが図7である。バッファがない場合には各段のサイクルタイムがそのまま滞在時間に反映されるので、図6と同じ傾向のグラフになっている。しかし、滞在時間には前段の処理が長いことによるアイドル状態の影響は表われないので、サイクルタイムの場合に比べればNの違いによる差は小さい。

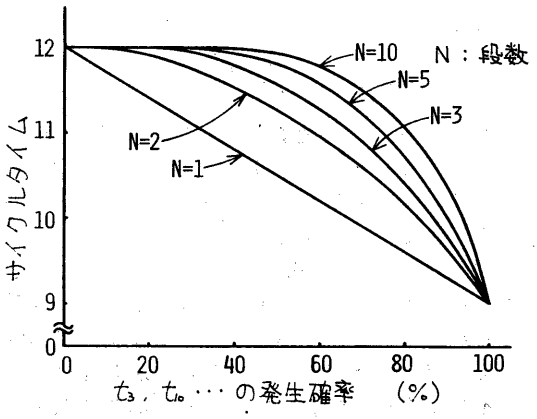

図6. 段数の違いによるサイクルタイムの変化

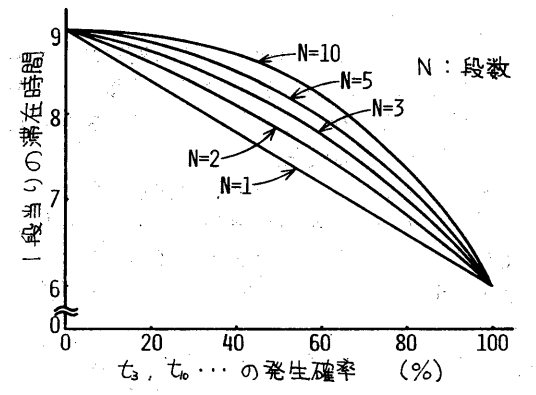

図7. 段数の違いによる滞在時間の変化

５．２．２ バッファのある場合

処理要素が５段（Ｎ＝５）の場合に、各段の間に入れるバッファの数を変えてシミュレーションした結果を、図８、図９、図10に示す。

バッファを入れることによって各段での処理時間のばらつきがある程度吸収されるので、サイクルタイムが減少することが図８からわかる。

しかし、バッファを入れることによって滞在時間は大きく増加する。この場合の滞在時間とは、１単位のプログラム＋データが１段目で受信されてから５段目で送信が完了するまでの時間である。この滞在時間は、バッファが２つの場合にはバッファがない場合の２倍近くにもなる。

スループットは

$$\text{スループット} = \frac{\text{処理量}}{\text{サイクルタイム}}$$

として計算し、処理量は１段での処理時間に等しいものとした。バッファによるスループットの改善が見られるが、これはサイクルタイムの改善によるものである。

以上から、各段の間にバッファを入れることでサイクルタイム、スループットが改善されることがわかった。反面、バッファによって滞在時間が伸びることもわかった。しかしながら、サイクルタイムの減少はアイドル時間の減少を意味し、資源の利用率の向上を表わすので、その点からもバッファの導入は有効なものであるということができる。

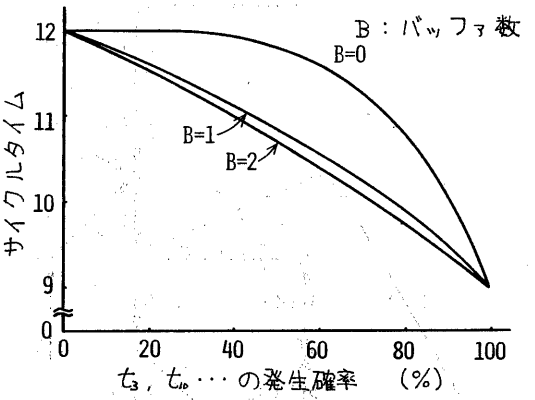

図８、５段の場合のバッファ数の違いによるサイクルタイムの変化

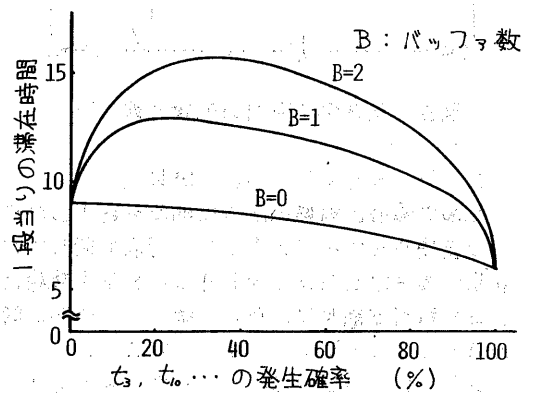

図９、５段の場合のバッファ数の違いによる滞在時間の変化

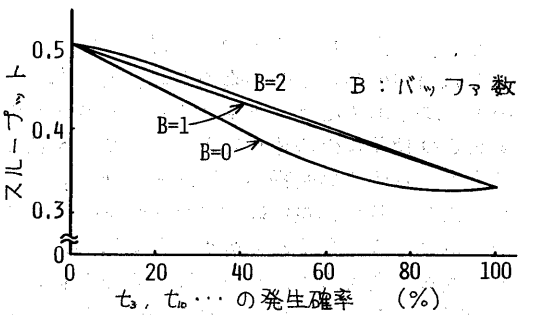

図10、５段の場合のバッファ数の違いによるスループットの変化

６、まとめ

知能リンクでは各処理要素での処理時間が異なるためにシステムの性能が低下する。その低下の様子、及び処理要素間にバッファを入れた場合の改善を解析した。

タイムペトリネットは時間の記述を含むペトリネットであるが解析が難しいため、制限を加えたタイムペトリネットを用いて知能リンクを簡単にモデル化し、シミュレーションを行なった。

それによって、処理要素間のバッファによりサイクルタイム、スループットが改善され、システムの性能が向上することを確認した。

参 考 文 献

(1) 中村維男、重井芳治："新しい伝送処理システム網の提案とその処理機構"，信学技報，CS78-80 (1978-08)

(2) Nakamura,t. and Shigei,Y:"An Effect of Employing Inteligent Links into Data Networks," in Proc. Int. Conf. Commun., pp.50.4.1-50.4.5,June 1980

(3) Peterson,J.L.:Petri Nets,Comput. Surv., Vol.9,No.3,pp.223-252 (Sept. 1977)

(4) Ramchandani,C.:"Analysis of asynchronous concurrent systems be timed Petri nets", MAC-TR-120,MIT 1974

(5) Zuberk,W.M.:"TIMED PETRI NETS AND PRELIMINARY PERFORMANCE EVALUATION", The 7th Annual Symposium on COmputer Architecture CH1494-4

(6) Merlin,P.M.:A Methodology for the Design and Implimentation of Communication Protocols",IEEE Trans. on COM Vol.COM-24,No.6 pp.614-621,June 1976