分析系统性能的一种方法：时间Petri网

林伟国　严隽永（海军工程学院）

摘　要

本文利用时间Petri网，讨论了分析系统性能的一般方法。在此基础上，进一步讨论了局域网中的CSMA/CD方式，并将此方法扩展到令牌总线方式上，为分析系统的物理特性和结构设计提供了一种方法。

1．时间Petri网

一般概念上的Petri网只对系统的逻辑特性进行分析，而不考虑系统的时间特性及性能。时间Petri网概念是七十年代提出的，它对系统的设计者来讲是相当重要的，因为它考虑了时间这个因素，可以形式化地讨论系统的一些性能，并可证明系统的安全性和活跃性[2]。

Sifakis提出并给定的时间Petri网具有如下的特性[4]：

时间Petri网(TPN)是一个六元组 $N=(P,T,\alpha,\beta,\tau,V)$，其中：

P：位置的集合，$P\neq 0$

T：变迁的集合，$T\neq 0$

α：$P\times T\longrightarrow I_N$(整数)，为向前相关函数

β：$P\times T\longrightarrow IN$(整数)，为向后相关函数

τ：$(\tau_1,\tau_2,\cdots,\tau_i,\cdots)$，为一实数递增序列，称为实数基

V：$P\times T\longrightarrow\tau$，使得$\forall(P,\tau_j)\in P\times\tau$：$V(P,\tau_i)=\tau_j\Rightarrow\tau_j>=\tau_i$

对于TPN也有相应的Petri网图，用圆圈表示位置，而用短线表示变迁。在Petri网图中，从位置p_s到变迁t_j之间存在一有向弧当且仅当$\alpha(p_s,t_j)=n(s,j)\neq 0$。而从$t_r$到$p_w$存在一有向弧当且仅当$\beta(p_w,t_j)=n(w,r)\neq 0$。标记M将位置P中是否存在标志映射为整数：$P\xrightarrow{M}IN$(整数)。下面是几条TPN规则：

a)在TPN中的标志呈两种情况之一：可行或不可行。初始时，每一位置p含有$M_0(p)$个可行标志。

b)变迁t是可发射的当且仅当连向t的每一个p_i，至少含有$\alpha(p_i,t)$个可行标志。

c)变迁t的发射在允许的情况下可以连续不断地发生。变迁t的发生从p_s中将移出$\alpha(p_s,t)$个可行标志，p_w中将接收到$\beta(p_w,t)$

很好地适合其他容易用比较少量的不同对象类定义的问题，许多模拟问题都有这些性质。

支持具有大量不断变化的对象类的研究工作不是我们的目标。因为不论何时要加入新的类或方法选择器，整个程序必须重新编译，这将是十分麻烦的，这些改变暗示着要重新定义全部数据类型。

这一方法学需要程序员注意该程序设计语言未提供的机制。Pascal根本不提供封装机制：由于必须通过全程变量访问对象，所以所有代码总是能访问所有的对象。实际上，我们发现在复合对象中不经消息传递机制，从顶层对象的域读取数据，子对象的方法是有用的。这在面向对象的语言中是不可能的。在我们的方法学中，这只是由程序员的喜好和斟酌来实现的。

（参改文献略）

〔沈国沛　译自《Communication of the ACM》，sept. 1987, Vol.30, No.9 p772—776〕

个可行标志。

d)位置p_i在它得到标志的时刻τ_i和使它变得可行的时刻$v(p_i,\tau_i)$期间保持不可行的状态。这样在每个位置p_i标志被延迟了z_i时间：

$$V(p_i,\tau)\in P\times\tau;\ V(p_i,\tau)-\tau=z_i=\text{constant}$$

对于具有n个位置的TPN，向量$O^t(\tau)$被定义为：$O^t(\tau)=[g_1(\tau),g_2(\tau),\cdots,g_n(\tau)]$。$O^t(\tau)$被称为时间$\tau$的负荷变量。每一元素$g_i(\tau)$表示在时刻$\tau$时$p_i$的标志。

对于间隔$\Delta\tau=\tau-\tau_0$，网中的负荷变量的平均变化可由向量给出：$I(\tau)=O(\tau)/\Delta\tau$。称$I(\tau)$为现行向量。向量I的每一元素$i_k$表示了在$\Delta\tau$间隔内变迁$t_k$发生的平均频率。

Sifakis已证明：为了保证网中总的负荷变量有界，必须使对应各变迁的i_k为常数，这是由于网的周期发生原因。对应于周期发生的最大可计算率可由解一组线性方程给出：

$$CI=0 \qquad I>0 \qquad (1)$$

$$\{J_i|Q(\tau_0)>=J_i|ZC^tI\} \qquad (2)$$

在此C=TPN的关联矩阵$[c_{ij}]_{n\times m}$

$$c_{ij}=\begin{cases}\beta(p_i,t_j) & \text{如果}\beta(p_i,t_j)\neq 0\\ -\alpha(p_i,t_j) & \text{如果}\alpha(p_i,t_j)\neq 0\\ 0 & \text{否则}\end{cases}$$

$C^t=[c_{ij}]_{n\times m}$

$$c_{ij}=\begin{cases}\beta(p_i,t_j) & \text{如果}\beta(p_i,t_j)\neq 0\\ 0 & \text{否则}\end{cases}$$

$\{J_i\}$是I的产生因子，I是$CX=0$的一组非负解。$Q^t(\tau_0)$是初始标记向量。Z是一个延迟对角矩阵。$Z=\text{diag}[z_1,z_2,\cdots,z_n]$。

如果(2)变成等式，则称网在功能上以自然的速率产生。

以上的线性方程在TPN可被分解为子网的情况下可很方便地求解。

2. 总线控制方式的性能分析

局域网中的总线控制方式分为CSMA/CD和令牌总线两种基本模式，其单总线时间Petri网分别见图1和图2。

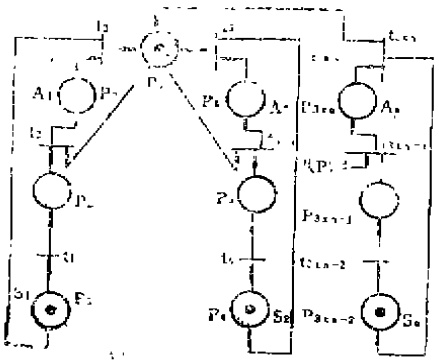

图1 CSMA/CD的时间Petri网图

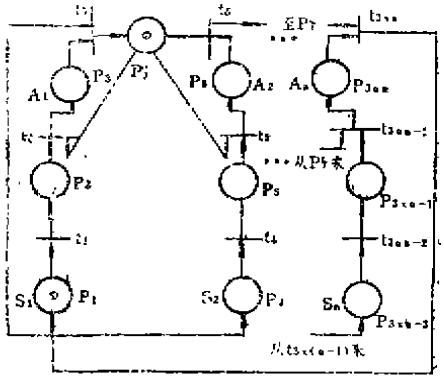

图2 令牌总线的时间Petri网

这里P_1，P_4，…，$P_{3\times n-2}(n\geq 1)$：表示节点机正在处理内部信息。P_3，P_6，…，$P_{3\times n}$ $(n\geq 1)$：表示节点机访问总线。P_7'：表示总线可行。P_2，P_5，…，$P_{3\times n-1}(n\geq 1)$：表示节点机为访问总线而必须等待的延迟。t_1，t_4，…，$t_{3\times n-2}(n\geq 1)$：表示待发信息已就绪并发出总线请求。t_3，t_6，…，$t_{3\times n-1}(n\geq 1)$：表示总线允许。t_3，t_6，…，$t_{3\times n}(n\geq 1)$：表示访问结束。

对以上单总线的Petri网图只要增加P_7'的标志数便可得到CSMA/CD方式的多总线Petri网图。

在此，我们用P_1，P_4，…，$P_{3\times n-2}(n\geq 1)$来表示节点机的状态。如果这些节点机是活跃的，则相应位置就存在有标志。在此假定每个节点机处理内部信息的时间用$S_1,S_2,\cdots,S_n$来表示。S_i的值根据需完成的任务所花费的时间而定。

P_2, P_6, …, P_{3n-1} $(n\geqslant 1)$表示相应的节点机访问总线的状态，即发送信息。对于CSMA/CD和令牌总线而言，每条总线每个时刻只能由一个处理机占用。假定节点机占据总线的时间分别为A_1, A_2, …, A_n。A_i值的大小依所发信息的长度而定，其中含协议中用于出错控制和出错后恢复的时间。为简单起见，假定A_i表示节点机i的平均访问总线时间。

P_7表示总线是可用的或不可用的状态。对于单总线而言，只要有一个节点机占据总线，则总线便处于不可用状态。对多总线而言，CSMA/CD方式允许在剩余的总线中再次竞争以便发送信息。P_2, P_5, …, P_{3n-1}表示的仅是一个伪状态。

t_1, t_4, …, t_{3n-2} $(n\geqslant 1)$表示节点机发出总线请求命令。当节点机完成对待发信息的处理时，便可发出此请求。t_2, t_5, …, t_{3n-1} $(n\geqslant 1)$表示节点机已获得总线，可以进行信息的发送了。而t_3, t_6, …, t_{3n} $(n\geqslant 1)$表示发送完毕。对于CSMA/CD，所有的节点机都可投入到再次竞争总线的过程中。而对于令牌总线，仅仅是逻辑环中的下一节点获得发送信息的权力。

利用[5]中的算法可证明该Petri网是活跃及非死锁的。在此，我们首先分析一下CSMA/CD的性能。从图1中可得出：

$$Z=\mathrm{diag}[z_1\ 0\ z_3\ z_4\ 0\ z_6\ z_7\cdots z_{3n}]$$

其中，$z_1=S_1$, $z_3=A_1$, $z_4=S_2$, $z_6=A_2$……

$$C_{(3n+1),(3n)}=\begin{vmatrix} -1 & 0 & 1 & 0 & 0 & 0 & \cdots\cdots \\ 1 & -1 & 0 & 0 & 0 & 0 & \cdots\cdots \\ 0 & 1 & -1 & 0 & 0 & 0 & \cdots\cdots \\ 0 & 0 & 0 & -1 & 0 & 1 & \cdots\cdots \\ 0 & 0 & 0 & 1 & -1 & 0 & \cdots\cdots \\ 0 & 0 & 0 & 0 & 1 & -1 & \cdots\cdots \\ 0 & -1 & 1 & 0 & -1 & 1 & \cdots\cdots \\ \vdots & & & \vdots & & \vdots & \end{vmatrix}$$

$$Q^i(\tau_0)=[1\ 0\ 0\ 1\ 0\ 0\ 1\cdots\cdots 1\ 1]$$

$(3\times n)+1$个元素

解方程$CI=0$，可发现$I=[i_1i_1i_1i_2i_2i_2\cdots\cdots i_n i_n i_n]$

$$J_1^t=[1\ 1\ 1\ 0\ 0\ 0\cdots\cdots 0]$$

$(3\times n)+1$

$$J_2^t=[0\ 0\ 0\ 1\ 1\ 1\ 0\cdots\cdots 0]$$

$$\vdots \qquad\qquad \vdots$$

$$J_n^t=[0\ 0\ 0\ 0\ 0\ 0\ 0\cdots\cdots 1\ 1\ 1]$$

$(3\times n)+1$

$$J_{n+1}^t=[0\ 0\ 1\ 0\ 0\ 1\ 1\cdots\cdots 0\ 0\ 1]$$

根据(1)和(2)，便可得i_1, i_2, ……, i_n的最大值满足下列条件：

$$1>=i_1(S_1+A_1)$$
$$1>=i_2(S_2+A_2)$$
$$\vdots \qquad\qquad \vdots$$
$$1>=i_n(S_n+A_n)$$

$$1>=\sum_{p=1}^{n}(A_p+z_7)\cdot i_p \qquad (3)$$

设d_i表示在每个循环中得到总线所产生的延迟，故

$$i_1=1/(S_1+A_1+d_1)$$
$$i_2=1/(S_2+A_2+d_2)$$
$$\vdots \qquad\qquad \vdots$$
$$i_n=1/(S_n+A_n+d_n)。$$

i_i表示节点p_i在循环时间T中的最大可能速率。在此T定义为总线应答的完整周期的时间(对每一个不同的节点机取最大的时间)。

总线的信息吞吐率M_i的大小决定于每一个S_i, A_i和d_i的值。现假设S_i和A_1, d_i的值最大，故$T=T_1=S_1+A_1+d_1$。

设节点机P_i在T时间内可完成m_i次信息发送的任务$(i=2, \cdots, n)$。

即$i_1=i_2/m_2=i_3/m_3=\cdots=i_n/m_n$

因此$1>=\sum_{p=1}^{n}(A_p+z_7)\cdot i_p$便可化为

$$1>=\sum_{p=1}^{n}(A_p+z_7)\cdot m_p\cdot i_1(m_1=1)$$

故 $M_i=(1+m_2+\cdots+m_n)/n\cdot T=(1+$

(转第39页)

以用任何语言实现，如Darwin程序设计环境[2]完全采用PROLOG实现，能否用OOPL来实现OOLGS，使有关OOP的基本功能依然由语言本身支持，法则部分只是在其上再追加OOP的其它机制(如多重继承，不完全继承等)，这样可以在保持LGS优点的基础上改善其环境效率。

3.LGS和规则系统(rule-based system)，约束系统(constraint-satisfaction system)既有联系又有区别，联系是它们都涉及到用规则(rule)描述问题，而区别在于后两者的目的是要取代(replace)整个系统，而LGS只是希望其中的法则能制约(regulate)系统的结构、功能及其演变，便于人们对大型系统的开发和管理。

4.LGS除描述OOP语义外，还可以控制原型系统(prototypical system)中的消息授权(delegation)处理[8]；或用一组控制模块间访问法则来构成MIL[2]；或利用LGS来描述资源的存取保护机制等，这些都是LGS中需要进一步考虑的课题。

参考文献

〔1〕 Goldberg,A. and Robson, D.,Smalltalk-80: The Language and its Implementation, Addison-Wesley, 1983.

〔2〕 Minsky, N. H. & Rozenstein, D., A Software Development Environment for Law-Governed Systems, In Proceedings of ACM/SIGSOFT Symposium on Software Development Environments, pp65-75, Nov.1988.

〔3〕 Naftaly H. Minsky & Rozenshtein,D., Controllable Delegation: An Exercise in Law-Governed Systems. In Proceedings of the OOPSLA's 89 Conference, pp 371-380, Oct. 1989.

〔4〕 Peter Wegner, Dimensions of Object-based Language Design, In Proceedings of the OOPSLA'87 Conference, pp 168-181, Oct. 1987.

〔5〕 陈火旺等，程序设计方法学基础，湖南科学技术出版社，1987.

〔6〕 Brad, J. Cox, Object-Oriented Computing: An Evolutional Approach Addsion-Wesley, 1986.

〔7〕 Jay Almarode, Rule-Based Delegation for Prototypes, In Proceedings of the OOPSLA'89 Conference, pp 363-370, Oct. 1989.

（接第51页）

$$+\sum_{i=2}^{n} m_i)/T \cdot n \qquad (4)$$

方程(4)的最大值是一个背包问题，即是在最小的T时间时$\sum_{i=1}^{n} m_i$为最大，可用相应的算法求解出(4)的最大值，即总线的最大可利用率。从中可知，要想提高总线的吞吐率，须使总线的使用有选择性。

当考虑的情形为$z_7=0$时，即后一节点机在前一节点机刚释放总线时便可获得总线的使用权，即有

$$1 > \sum_{p=1}^{n} A_p \cdot i_p$$

这时总线的利用率提高了$T/(T-z_7)$。

对于多总线，设有b根，则(3)式可化为

$$b > = \sum_{p=1}^{n} (A_p + z_7) \cdot i_p$$

随着b的增大($b=<n$)，i_p便接近于最大允许值$1/(S_p+A_p)$。

对于令牌总线的分析可循此法进行。

3. 结 论

本文利用时间Petri网作为工具讨论了局域网中的两种总线控制方式及相应的系统性能。从中可看到，Petri网不但可以作为一种论证工具，同时也是一种较优的评价工具。用时间Petri网作为性能评价工具，可很方便地和计算机工具结合起来，为设计和评价一个系统提供了一种方法，所以说时间Petri网是一个值得注意的研究领域。(参考文献略)