基于网格网络的十字仲裁集互斥算法*)

王 征[1,2] 刘心松[2]

(西南财经大学经济信息工程学院 成都 610074)[1] (电子科技大学 8010 研究室 成都 610054)[2]

摘 要 分布式互斥是网格分布式系统的重要问题。根据网格系统的特点,提出了新型的分布式互斥算法。该算法基于网格网络的行列生成分布式互斥十字仲裁集;采用 Lamport 逻辑时戳保证消息的时序性;算法采用"探测"消息进行系统的容错处理。分析与仿真证明,该算法具有较低的消息复杂度、较短的响应延迟以及较好的容错性能。

关键词 分布式互斥,网格,仲裁集

Mesh Network Distributed Mutual Exclusion Algorithm Based on Cross Quorums

WANG Zheng[1,2] LIU Xin-song[2]

(CCSE, Southwestern University of Finance and Economics, Chengdu 610074, China)[1]

(8010 R&D CCSE, University of Electronic Science and Technology, Chengdu 610054, China)[2]

Abstract Distributed Mutual Exclusion (DME) is an important problem of distributed mesh systems. According to the properties of mesh systems, a novel algorithm MNCME was presented for them. Based on the rows and columns, the algorithm generated distributed mutual exclusion cross quorums. And Lamport's logical timestamps were utilized to guarantee the time sequence. Furthermore, "Probe" messages were employed to implement the fault-tolerance of the algorithm. Analysis and simulation results show that it has lower message complexity, shorter response delay and better fairness than the traditional algorithms do so.

Keywords Distributed mutual exclusion, Mesh, Quorum

1 引言

随着大规模生物及化学计算、天气预报、石油勘探等行业对超级计算能力的需求增长,传统的超级计算工业领域出现了复苏的趋势。群集系统的高可扩展性和高性价比,使它成为与网格技术并驾齐驱的超级计算技术之一,受到了超级计算工业领域的普遍重视[1,2]。但目前对群集系统的研究主要集中于应用算法的实现,对提高群集系统本身的理论研究还不够全面。例如网格(注意:此处的网格是英文 mesh 的翻译,与现在流行的网格 Grid 有区别)环境下的分布式互斥算法即是一例。网格网络中的节点往往需要并发地访问一个共享资源,而该资源只能被一个节点使用。访问该资源的代码段被称为临界区(CS)。如何保证节点互斥地使用临界区、缩短进入临界区的响应时间、减少互斥操作所需的消息数等问题,是分布式互斥算法所研究的重点。在分布式互地斥算法的研究过程中,Lamport, Richard&Agrawala 和 Maekawa 等人对全分布互斥算法做出了里程碑式的贡献[3]。其中,Lamport 提出了逻辑时戳(Logical Timestamp)的概念,采用该时戳后,在没有全局物理时钟的全分布式系统中,消息可以根据逻辑时戳表示的优先级进行全局排序。Richard&Agrawala 巧妙地利用延迟应答的方式将系统中各进程的资源请求按逻辑时戳组成一个动态令牌环,该算法中应答与释放消息合而为一,将算法具有在 N 个节点的系统中的进入一次临界区所需消息数降为 $2(N-1)$。Maekawa 提出了仲裁集(Quorum)的概念[4],并提出了基于有限投影平面的仲裁集构建方法,消息数下限为 $3m-5m(M=\sqrt{N})$,从而将算法的消息复杂度降为 $O(\sqrt{N})$。传统的 Maekawa 算法,单个节点不用请求系统中所有节点的许可,只需要请求一个节点请求集的许可。该算法的请求集需要满足以下条件:这里设网格系统共有 N 个节点 $\{p_1, p_2, p_3, \cdots, p_N\}$;仲裁集为 $QS=\{S_1, S_2, S_3, \cdots, S_k\}$,其中 $K=\sqrt{N}$,S_i 中包含 $\sqrt{N}$ 个节点,则

(1) $\forall S_i, S_j \subset QS \quad S_j \cap S_i \neq \emptyset$ 相交性;

(2) $\forall i \leqslant N \quad P_i \in S_i$ 等值性;

(3) $\forall S_i, S_j \subset QS \quad \forall i \neq j \quad S_i \not\subset S_j$ 最小性。

其中,仲裁集只要满足(1)(2),就能够保证了互斥的正确性。在竞争发生时,相交区域的节点作为仲裁节点决定请求节点进入临界区域的顺序;(3)主要是为了提高系统性能,而非必要条件。目前,研究人员提出了一些仲裁集的构建算法,但是这些算法通常都是基于系统的逻辑结构,而非物理结构;脱离系统实际拓扑结构的算法设计将导致互斥操作的低响应率和高消息复杂度,进一步导致系统性能下降。

本文提出了基于十字仲裁集的网格网络分布式互斥算法(MNCME: Mesh Network Distributed Cross Mutual Exclusion)。第 2 节将介绍算法的主要思想,并给出证明;同时,给出算法采用的数据/消息结构;最终给出算法描述。第 3 节将从理论上分析算法的性能,主要包括消息复杂度、响应延迟等。第 4 节将给出 MNCME 的仿真实验结果,并与其他算法作对比。最后总结全文。

2 MNCME 算法

本节首先给出算法的主要思想,并说明其可行性;其次给出算法所需的数据结构和消息结构;最后给出算法实现的伪

*)本文获四川应用基础研究项目(04JY29-017-2)资助,获科技型中小企业技术创新基金(04C26225110223)资助。王 征 博士,主要研究方向为分布式算法等;刘心松 教授,博导,主要研究方向为分布式并行等。

代码。

2.1 算法思想

MNCME算法充分考虑了网格(Mesh)系统的实际拓扑结构对分布式互斥算法带来的影响:网格系统中非边缘节点通常有四条链路,边缘节点有2～3条不等;所有节点构成一个$N\times M$的矩阵;节点相互路由信息,采取存储转发通信模式。它的这些特性给传统分布式互斥算法的实施带来很多困难。

首先,网格系统采取存贮转发通信模式和节点非直接相连的拓扑结构,而传统的分布式互斥算法通常假设所有的节点间都采用直接相连的通信模式。网格系统中,两节点间一条消息的传递,可能包括多条存储/转发消息的生灭,因此传统算法不考虑节点间距离以及采用点到点通信的方式将导致算法性能在网格系统中严重下降。例如,传统的Maekawa算法在构建仲裁集时,不考虑节点的物理位置,则仲裁集内部通信将激增。

其次,如前所述,网格系统是具有静态的、整齐的网络拓扑结构,因此节点可以通过计算获取自身位置、通信目的地距离等信息,而不用从网络中搜索。为满足最小性,传统算法通过广播/组播等主动搜索在每个节点中建立仲裁集的方法将导致系统的网络流量增加。

MNCME算法的主要思想基于网格的如下两个重要性质:

(1)网格网络中的每个节点至少属于一行和一列。

(2)网格网络中的任意一对行列必然相交。

其中第一条性质符合仲裁集的等值性;第二条符合仲裁集的相交性。因此,网格系统可以通过选择节点所处的行列组成十字仲裁集。

下面引入这样的例子:

(1)当两个节点P_i、P_j都要进入临界区时,它们向自己所属的网络行列上的边缘节点发送请求消息;消息途经的各个中间节点将这些请求消息插入请求队列。

(2)因为任意从属于不同节点的行列必然相交于节点(仲裁节点)P_k,该节点将会接收到2个不同的请求;仲裁节点对不同的请求进行排序,向高优先级节点发送应答。

(3)高优先级节点P_i接受到所处的行和列上所有节点的应答后,进入临界区。

(4)高优先级节点P_i在退出临界区后,向低优先节点P_j发送应答,使之进入临界区。算法的实例见图1。

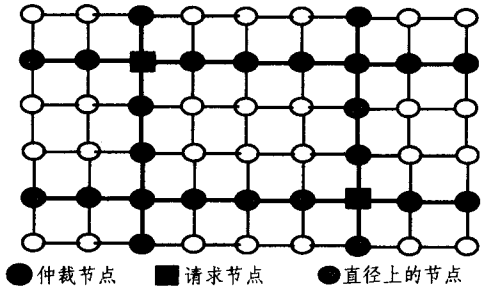

图1 MNCME算法实例

2.2 定义

本节给出分布式互斥模型的定义,以便于概念化、形式化的描述算法;其中请求R_i的优先级定义为$Pr(R_i)$。

定义1(并发(Concurrent)) R_i和R_j是并发的,当且仅当R_i在R_j的生存周期内产生或者R_j在R_i的生存周期内产生,这个定义是对称的[5]。

定义2(并发集ConcurrentSet) $(Cset)$:$Cset_i=\{R_j \mid R_j$ is concurrent with $R_j\}\cup\{R_i\}$

定义3(并发集首(ConcurrentHeader)) $(Cheader_i)$:$R_i:Pr(R_i)>Pr(R_j)\ \forall R_i,R_j\in Cset_k\ i\neq j$

定义4(并发集尾ConcurrentTailer)

$(Ctailer_i)$:$R_i:Pr(R_i)<Pr(R_j)\ \forall R_i,R_j\in Cset_k\ i\neq j$

定义5(前驱Pred) $Pred(R_i,Cset_k)=R_j$ iff $R_j\in Cset_k \wedge Pr(R_i)<Pr(R_j)\wedge\neg(\exists R_k\in S\mid(Pr(R_i)<Pr(R_k)<Pr(R_j)))$

定义6(后继Succ)

$Succ(R_i,Cset_k)=R_j$ iff $R_j\in Cset_k\wedge Pr(R_i)>Pr(R_j)\wedge\neg(\exists R_k\in S\mid(Pr(R_i)>Pr(R_k)>Pr(R_j)))$

2.3 数据结构与消息

本算法中消息的统一格式如下:msg_name(sender, receiver, parameter)。sender为消息发送者;receiver为消息接收者;parameter为消息参数列表。算法中采用的消息及其发送方式如下。

Request$(i,path,timestamp)$:请求消息,用于节点P_i向所处的行列之合path上所有的节点请求进入临界区。*timestamp*是消息携带的Lamport逻辑时戳。值得注意的是:该消息在多跳网络中缓存/转发,但是并不改变逻辑时戳值和转发节点计数器的值,从而保证事件排序的正确性。此外,该消息通过进程间通信发送给自身所在节点,而不通过网络。Path列表用于记录应答消息转发时通过的路径。Probe消息,参数与前者一致,用来"探索"节点所属的行列的边缘节点。

Reply$(i,j,Path)$:应答消息,用于P_i节点传递本行或者本列的应答给P_j节点。通常处于行列中的节点,需要接受4条应答消息,方可进入临界区。处于网格边缘的节点,通常仅需要接受2至3条应答消息就可以进入临界区。Path的定义同上。

Release$(i,timstamp)$:释放消息,用于节点P_i向所处行列上所有的节点表明自己已经退出临界区,允许其他节点进入。

Failed、Inqurie和Relinquish消息同Maekawa算法中的一致,限于篇幅原因,不做累述;读者可以参见文献[3]。

算法采用的数据结构的命名格式为Variable_Name$_i$,其中i是拥有该数据结构的节点标识。算法中具体的数据结构如下。

用于实现Lamport逻辑时戳的整型变量:LC_i(*Local Clock*),代表Lamport算法的本地计数器,MS_i代表接收到的最大时戳。关于Lamport的详细算法与证明,参见文献[4]。特别注意,算法中的优先级大小与Lamport时戳大小是反序的关系。

请求队列RQ_i:用于保存接收到的请求消息。与传统算法不同,该队列长度不固定,同时该队列中的元素按照时戳确定的优先级进行排序。

十字仲裁集D_Quorum_i,用于保存节点P_i所处行列包含的节点。

2.4 算法描述

算法的实现采用消息/事件驱动方式描述,即当进程接收到特定消息或者进程中发生特定事件时,进程调用规定过程。MNCME算法中的过程名按照"On_EventName"形式定义,EventName代表被触发的事件名。P_i进程中的相应过程如

下：

```
Procedure On_InitVariable;// Pi 初始化变量
  LCi:=0; //时戳变量初始化
  Has_Tokeni:=False;
  //D_Quorumi 通常在启动前设置好
End;
Procedure On_ReqCS; // Pi 请求进入临界区
  Requesti:=(i,D_Quorumi, LCi); //填充请求消息
  Insert Ri into RQi;//将 Ri 插入请求队列
  Send Requesti to the D_Quorumi; //发送请求
  While (Eof of RQi) do
  If Pr(Rj)>Pr(Ri) then delete Rj from RQi;
  //清除优先级高于 Ri 请求的 RQi 元素
End;
Procedure On_RcvRequest; // Pi 接收到 Pj 请求 Rj
If ( Pi in Rj->path) then
{
  LCi:=1+MAX(LCi,Rj. timestamp); //修正时戳
  Insert Rj into RQi by the order LCi;
  If Rj is the CHeader of Cseti then
    If ( Pi is the last of Rj->path) then //判断是否边缘节点
      Send Reply(i, j, Path) to Pj; //应答
  Else Transmit Rj to Next(Rj->path); //转发
}
End;
Procedure On_RcvReply;// Pi 接收到应答消息
    If ∑Replyj== D_Quorumi then //应答组成仲裁集
      Pi enters into the CS;
End;
Procedure On_RcvRelease;// Pi 接收到释放消息
    Delete Rj from RQi
    If RQi is not NULL then //转发应答
      Transmit the Rk of the CHeader of Cseti;
End;
Procedure On_ExCS; // Pi 退出临界区
  Send Releasei to the D_Quorumi; //发送释放消息
End;
```

Failed、Inqurie 和 Relinquish 的接收与发送事件定义同 Maekawa 算法一致，限于篇幅原因，不做累述，有兴趣的读者可以参见文献[4]。在规定延迟内，Reply 消息未返回，请求节点释放 Probe 消息，探测并生成一条新的到达边缘节点的路径，从而保证算法在异常情况下能够正常运行，提高了算法的容错能力。

3 性能分析与仿真

3.1 性能分析

消息复杂度(完成一次互斥操作所需的消息数量)是衡量分布式互斥算法的重要指标。传统的消息复杂度度量方法通常假设网络中的所有节点都是直接相连，而网格网络具有多跳性，因此传统方法并不适合度量其消息复杂度。下面详细分析 MNCME 在网格网络中的消息复杂度。如第 2 节所述，由于该算法修改了 Request 消息结构，使得节点无需从自身出发遍历每个节点，大大减少了所需要的消息量。这里按两种情况分析所需的消息量。

轻载环境下，网络中不存在节点竞争，此时 MNCME 算法仅需要发送 Request 和接收 Reply 消息即可进入临界区。请求进入临界区的节点首先发送 2 或 4 个请求。这些请求消息在网络不断被存贮/转发，直到遍历所处的十字仲裁集。此过程中共有 $N+M-2$ 个消息生灭。行列两端节点在确认该请求节点的优先级最高后，发送应答消息给它，这同样是一个遍历特定行列的过程，此过程中同样将有 $N+M-2$ 个消息生灭。因此，为了单个节点进入临界区，共需要 $2\times(N+M-2)$个消息生灭。退出临界区时，MNCME 算法需要 Release 消息释放临界区，这同样是一个遍历特定行列的过程，此过程中同样有 $N+M-2$ 个消息生灭。因此，轻载环境下，为了单个节点进入临界区，系统中有 $3\times(N+M-2)$个消息生灭。

重载环境下，Failed、Inquire 和 Relinquish 在算法中得以使用。由于这些消息不需要遍历整条直径，消息的传递范围与请求节点到仲裁节点的距离有关，通常这三种消息的生灭会产生 0(仲裁节点就是请求节点本身)到 $N+M-2$ 个消息。因此，重载情况下，节点需要 $5\times(N+M-2)$个消息进行互斥。

尽管传统 Maekawa 仲裁集在理论上的消息复杂度为 $O(\log N)$，但实际上，由于一条消息反复在网格网络中转发/传递/生灭，通过了许多不属于请求节点所属的仲裁集的点，因此算法性能在实际网格网络环境中骤降。随机分布的 Maekawa 算法中，设网格网络中 $P[i,j]$是请求节点，$P[x,y]$是仲裁集里的节点，则 $S_k(k=i*N+j)$仲裁集完成一次互斥操作所需的消息总数为 $3\times(\sum Hop(P[i,j], P[x,y])$至 $5\times(\sum Hop(P[i,j], P[x,y])$，其中 $P[x,y]\in S_k$；由于网格网络的自身特性，节点 $P[i,j]$完成一次互斥操作所需消息数为 $3\times\sum(|i-x|+|j-y|)$到 $5\times\sum(|i-x|+|j-y|)$，其中 $P[x,y]\in S_k$。以拥有 3×4 个节点的系统中的 Maekawa 仲裁集为例，由于 Maekawa 仲裁集需要系统拥有素数个节点，因此本例采用冗余的方法，使 $P_1=P_{13}$，从而构建 Maekawa 仲裁集(尽管这种方法会带来仲裁集轻微地不平衡[2]，但是并不影响其正确性)：

$S_1=\{P_1, P_2, P_3, P_4\}$ $S_2=\{P_2, P_5, P_8, P_{11}\}$ $S_3=\{P_0, P_6, P_8, P_1\}$ $S_4=\{P_4, P_6, P_{10}, P_{11}\}$ $S_5=\{P_5, P_1, P_6, P_7\}$ $S_6=\{P_6, P_2, P_9, P_{12}\}$ $S_7=\{P_7, P_2, P_{10}, P_1\}$ $S_8=\{P_8, P_1, P_9, P_{10}\}$ $S_9=\{P_9, P_3, P_7, P_{11}\}$ $S_{10}=\{P_{10}, P_3, P_5, P_{12}\}$ $S_{11}=\{P_{11}, P_1, P_{12}, P_1\}$ $S_{12}=\{P_{12}, P_4, P_7, P_8\}$

此时，各仲裁集完成一次互斥操作需要的消息数分别为：18-30；24-40；18-30；30-50；12-20；18-30；21-35；33-55；36-60；21-35；27-45；18-30；这些数字远远超过理想环境中的计算结果。很明显，在网格网络中，*MNCME* 算法的消息复杂度大大低于随机分布的仲裁集算法。

时间复杂度也是衡量分布式互斥算法的重要指标之一(习惯上假设任意相邻两点间转发一条消息的平均通信时间为 T，执行一次临界操作的时间为 E(通常假设其与 T 相等)，用以度量算法的延迟情况。MNCME 算法的消息复杂度与请求节点的位置休戚相关，通常是请求节点到达其所在的行列的边缘节点的最大值。因此，其时间复杂度在 $\mathrm{MAX}(row/2, col/2)\times T+E$ 和 $\mathrm{MAX}(row, col)\times T+E$ 之间。传统 Maekawa 算法由于其仲裁集中的节点随机分布，造成了时间复杂度较高，通常在 $\mathrm{MAX}(row/2, col/2)\times T+E$ 和 $(row+col)\times T+E$ 之间。上例中，根据 A. W. Fu 的时间复杂度理论[6]：随机分布的仲裁集的最大延迟为 5；而十字仲裁集的最大延迟为 4(四个顶点所属的仲裁集)，低于前者的最大延迟。随机分布的仲裁集的平均延迟达到 39/12，而十字仲裁集的平均延迟为 30/13。由此对比可以看出，MNCME 算法具有较低的时间复杂度。

3.2 仿真结果

系统仿真模拟了网格网络环境中 MNCME 算法和传统 Maekawa 算法的运行情况，其中 Maekawa 算法中的节点为随机散布。

图 2 是 MNCME 算法和 Maekawa 算法的消息复杂度对比。从图中可以看出，MNCME 算法具有较低的消息复杂度，这是因为它充分考虑了网格网络中节点位置对仲裁集的影响。同时，该算法改造了请求等消息，这种改造使得原本需要点对点逐次发送仲裁集中所有的消息仅发送给临近节点即

可。由于 Maekawa 算法将节点随机散布在网络中，请求节点需要逐个将消息发送给仲裁集，消息传播的多跳性导致该算法的消息复杂度激增。

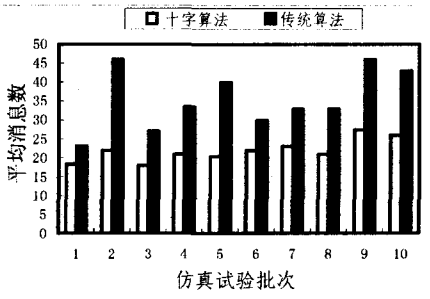

图 2　消息复杂度仿真结果对比

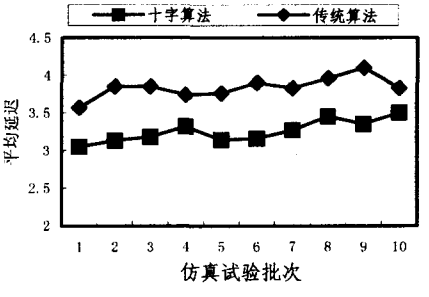

图 3　时间复杂度仿真结果对比

图 3 是 MNCME 算法和 Maekawa 算法的响应延迟对比。尽管传统 Maekawa 算法不需要遍历行列，但仲裁集随机散布在整个网络中，往往导致消息在线路上冲突，且随机扩散的仲裁集间距较大，因而使该算法在网格网络中的响应延迟反而比 MNCME 长。

结束语　本文根据网格网络的特点，提出了新型的基于十字仲裁集的分布式互斥算法 MNCME。该算法基于节点所处的行列生成十字仲裁集，并且用 Lamport 逻辑时戳保证消息的时序性。分析与仿真证明，和传统 Maekawa 算法相比，该算法具有较低的消息复杂度和时间复杂度。

参 考 文 献

[1] 舒继武，等. 大规模问题数据并行性能的分析[J]. 软件学报，2000，11(5)：628-633

[2] 黄铠，徐志伟. 可扩展并行计算技术、结构与编程[M]. 陆鑫达，等译. 北京：机械工业出版社，2000：145-223

[3] 尹俊文，邹鹏，等. 分布式操作系统[M]. 长沙：国防科技大学出版社，2001：68-82

[4] Maekawa M. A logN Algorithm for Mutual Exclusion in Decentralized Systems [J]. ACM Trans Computer Systems, 1985, 3 (2): 145-159

[5] 刘丹，刘心松. 基于读写特征的分布式互斥算法 [J]. 电子学报，2004，32(2)：326-329

[6] Fu A W. Delay-optimal Quorum Consensus for Distributed Systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8(1): 59-69

(上接第 193 页)

时，ARTMAP 神经网络更符合当前垃圾邮件特征不断变化的情况，更能适应环境的变化。

表 2　仿真结果

组别	输入	匹配模板	产生新模板
1	101		合法邮件
	110		垃圾邮件
	110		无法判定
	100		无法判定
2	101	合法邮件	
	110	垃圾邮件	
	100	无法判定	
	110	无法判定	
3	111	无法判定	
	110	无法判定	
	100	无法判定	
	110	合法邮件	

虽然 ARTMAP 神经网络过滤器能有效减少用户损失，但是不同的用户对同一封邮件有不同的偏好，如一封广告邮件，对需要它的人来说就是合法邮件，对不需要它的人，它就是一封垃圾邮件，这在 ARTMAP 过滤器中并没有体现出来。因此，如何实现在过滤过程中根据用户不同偏好，实现自动化、个性化的邮件过滤，将是下一步进行研究的重点。

参 考 文 献

[1] 刘洋，杜孝平，黄星华，等. 垃圾邮件的智能过滤系统设计探讨. 微机发展，2003，13(4)：1-3

[2] Gail C, Stephen G, John R. ARTMAP: Supervised Real-Time Learning and Classificaion of Nonstationary Data by a Self-Organizing Neural Network. Neural Networks, 1991(4): 565-588

[3] 潘文峰. 基于内容的垃圾邮件过滤研究. 学位论文. 中国科学院计算技术研究所，2004

[4] 庞剑锋，卜东波，白硕，等. 基于向量空间模型的文本自动分类系统的研究与实现. 计算机应用研究，2001(9)：23-26

[5] Yang Yi Ming, Jan P. A Comparative Study on Feature Selection on Text Categorization// International Conference on Machine Learning (ICML). 1997: 412-420

[6] Zhan Chuan, Lu Xianliang, Hou Mengshu, et al. A LVQ-based neural network anti-spam email approach. ACM SIGOPS Operating Systems Review, 2004: 35-39

[7] 吴跃，王佳. 基于 NB 的双级分类模型在邮件过滤中的研究. 计算机科学，2006，33(5)：110-112

[8] 赵治国，谭敏生，李志敏. 基于改进贝叶斯的垃圾邮件过滤算法综述. 南华大学学报(自然科学版)，2006，20(1)：33-38

[9] 成宝国，冯宏伟. 一个基于 Naive Bayesian 垃圾邮件过滤器的改进. 计算机技术与发展，2006，16(2)：98-99

[10] 刘震，周明天. 基于有监督 Bayesian 网络的垃圾邮件过滤. 计算机应用，2006，6(3)：558-561

[11] James C , Irena K , Josiah P. A Neural Network Based Approach to Automated E-mail Classification // Proceedings of the 2003 IEEE/WIC/ACM international Conference on Web Intelligence(WI'03). 2003: 702-705

[12] Cheepeng L, Jennhwai L, Kuan MeiMing. A Hybrid Neural Network System for Pattern Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005, 27(4): 648-653

[13] Dimitrios C, Michael G, Takis K. Classification of Noisy Signals Using Fuzzy ARTMAP Neural Networks. IEEE Transaction on Neural Networks, 2001, 12(5): 1023-1036

[14] Kiong L. Accurate and Reliable Diagnosis and Classification Using Probabilistic Ensemble Simplified Fuzzy ARTMAP. IEEE Transactions on Knowledge and Data Engineering, 2005, 17 (11): 1589-1593