# Petri 网与蚁群算法在 CDN 网络路由中的应用

叶剑虹[1,2],孙世新[1],张运生[1],周益民[1]
YE Jian-hong[1,2],SUN Shi-xin[1],ZHANG Yun-sheng[1],ZHOU Yi-min[1]

1.电子科技大学 计算机科学与工程学院,成都 610054
2.艾因霍芬科技大学 数学与计算机学院,荷兰艾因霍芬 P.O.Box 513,NL-5600 MB
1.School of Computer Science and Engineering,University of Electronic Science and Technology of China,Chengdu 610054,China
2.Department of Mathematics and Computer Sciences,Eindhoven University of Technology,P.O.Box 513,NL-5600 MB Eindhoven,The Netherlands
E-mail:leafever@163.com

**YE Jian-hong,SUN Shi-xin,ZHANG Yun-sheng,et al.Application of Petri nets and ant colony algorithm for solving routing problem on CDN.Computer Engineering and Applications,2008,44(19):31-35.**

**Abstract:** Content Delivery Network(CDN),helps to efficiently deliver the content from content providers to a large community of geographical distributed clients,calling for more effective routing algorithm with higher quality of QoS.This paper presents a new algorithm joint with the ant colony algorithm and Petri nets,used to solve the routing in stability of connected graph topology,considering with delay,delay jitter,bandwidth,packet loss and the least cost constraint.Simulation results show that this algorithm is reasonable and effective.

**Key words:** structured network;QoS;routing algorithm;ant colony algorithm;Petri nets

**摘 要:**CDN 让用户以最快的速度从最近的地方获得所需的信息,它对 QoS 路由提出了更高的要求,单纯的采用蚁群算法或是依靠 Petri 网模型中变迁发生寻径的方法都不能较好的解决 CDN 路由问题。基于稳定的 CDN 网络构建图状拓扑结构,提出了一种将 Petri 网与蚁群算法相结合,考虑多个路由限制的优化 QoS 路由算法。实验仿真表明,该算法能有效地求解 CDN 网络中的路由问题。

**关键词:**网络结构;服务质量;路由算法;蚁群算法;Petri 网
**DOI:**10.3778/j.issn.1002-8331.2008.19.009 **文章编号:**1002-8331(2008)19-0031-05 **文献标识码:**A **中图分类号:**TP393

## 1 引言

CDN[1,2]致力于内容和资源的分发,它由分布在不同区域的节点服务器群组成虚拟网络,通过动态部署网络内容到边缘,根据网络内容处理通信量,将访问请求转发给最优服务器,从而使用户能以最快的速度,从最接近用户的地方获得所需的信息。它对网络服务质量(QoS)提出了更高的要求。已证明当 QoS 约束条件包括两个或者两个以上的加法型度量时,该路由选择问题为 NP-hard 问题[3,4]。近年来,启发式蚁群算法(ACA)[5,6],因其具有正反馈,易于分布并行执行等特点成为解决该类问题的研究热点。已有文献[7]将其应用于 CDN 网络,取得了一定的成果。但蚁群算法也具有收敛速度慢,计算时间长,容易出现停滞现象(stagnation behavior)。在不知道路径所耗时间的情况下蚁群算法的引导方式只能是纯粹的信息素引导。

Petri 网[8-10],作为一种具有全局的并发、异步性的建模工具,其托肯的流动具有确定性,变迁的延迟信息是可知的。这使得依据 Petri 网的变迁发生寻找路由从一开始就有了搜索最短序列的倾向。将 Petri 网与蚁群算法相结合,能够为寻求高服务质量的 CDN 网络路由提供一种新的尝试。

本文将蚁群算法的选择概率公式做了必要的修改,从而可以灵活地进行蚁路引导,通过一定方式控制 Petri 网进行局部搜索。有利于加快路由搜索速度,同时 Petri 网托肯的流动性也防止了蚁群算法过早的陷入局部最优。实验仿真表明,该算法能有效地求解 CDN 网络中的 QoS 路由问题。另外,本研究还将 CDN 网络与自组织 P2P 网络相结合,构建了一种新型的网络模型,在效率及硬件开销方面取得较好的平衡。限于篇幅,这方面的研究工作将在另文中加以阐述。

**基金项目:**国家自然科学基金(the National Natural Science Foundation of China under Grant No.60473030);华为高校科技基金项目(the University Science Foundation of HuaWei Inc. under Grant No.YJCB2006057FT)。

**作者简介:**叶剑虹(1976-),男,博士生,主要研究方向为分布式并行计算,Petri 网理论及应用;孙世新(1940-),男,教授,博士生导师,CCF 高级会员,主要研究方向为分布式并行计算,网格技术;张运生(1972-),男,博士生,主要研究方向为 P2P 技术及视频编码;周益民(1980-),男,博士生,主要研究方向为视频编码,大规模并行计算。

**收稿日期:**2008-03-06 **修回日期:**2008-04-24

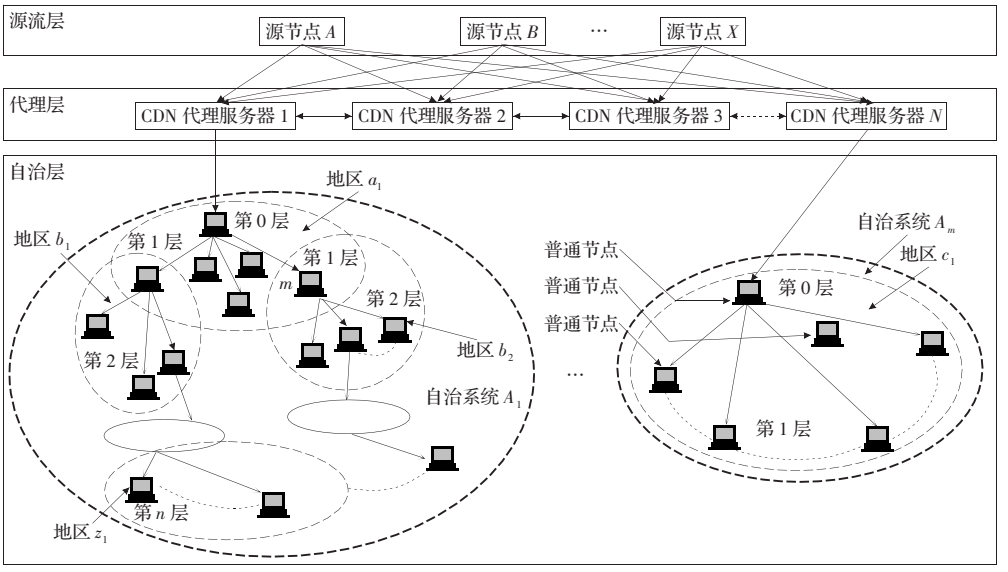



**图 1** P2P_CDN **网络拓扑结构图**

## 2 网络架构

鉴于描述上的完整性，文章给出了 CDN 网络与自组织 P2P 网络相结合的一个完整的网络逻辑结构。如图 1 所示，本文只关注 CDN 网络，即该模型中的源流层和代理层。提供内容的源服务器组成源流层，接收来自不同源服务器提供的共享资源的代理服务器组成代理层。其它层次的描述将在另文中加以阐述。本文假令 CDN 网络中的结点性能都稳定可靠，而且系统内一定存有需要的路由结点，对于无法搜索到的结点，很容易给出时限阈值。

## 3 算法描述

### 3.1 路由及结点的加入算法

以下仅对本节所需的一些基本概念作简单定义和描述。详细定义请参阅文献[5,6,8-10]。

设 $N=(V,E)$表示网络，其中 $V$ 表示网络结点的集合，$E$ 表示双向链路的集合。$s\in V$ 为源结点，$d\in V-\{s\}$ 是目的结点。(1) $\forall i\in E$定义 4 种度量参数，链路时延 $link\_delay(i)$、链路费用 $link\_cost(i)$、链路带宽 $link\_bandwidth(i)$、链路时延抖动 $link\_jitter(i)$。分别简记为 $l\_d(i)$、$l\_c(i)$、$l\_b(i)$、$l\_j(i)$；(2) $\forall j\in V$ 定义 4 种度量，结点时延 $node\_delay(j)$、结点费用 $node\_cost(j)$、结点时延抖动 $node_jitter(j)$和丢失率 $node\_lost(j)$。分别简记为 $n\_d(j)$、$n\_c(j)$、$n\_j(j)$、$n\_l(j)$。

**定义 1** 若 $p(s,d)$表示从源结点 $s$ 到目的结点 $d$ 的路由路径，其路径上的时延、带宽、时延抖动、包丢失率及费用分别如下式：$delay(p(s,d))=\sum_{i\in p(s,d)} link\_delay(i)+\sum_{j\in p(s,d)} node\_delay(i)$；$bandwidth(p(s,d))=\min_{i\in p(s,d)}\{link\_bandwidth(i)\}$；$jitter(p(s,d))=\sum_{i\in p(s,d)} link\_jitter(i)+\sum_{j\in p(s,d)} node\_jitter(i)$；$lost(p(s,d))=1-\prod_{j\in p(s,d)}(1-node\_lost)$；$cost(p(s,d))=\sum_{i\in p(s,d)} link\_cost(i)+\sum_{j\in p(s,d)} node\_cost(i)$。

对于单播路由请求 $R=(s,d,W_p,D_p,L_p,J_p)$，$s$ 是源结点、$d$ 是目的结点、带宽约束 $W_p$、时延约束 $D_p$、丢失率约束 $L_p$ 和时延抖动约束 $J_p$。若路由请求 $R$ 被接受，则路由选择算法找到路径满足下列约束条件：(1)时延约束：$delay(p(s,d))\leqslant D_p$；(2)带宽约束：$bandwidth(p(s,d))\geqslant W_p$；(3)时延抖动约束：$jitter(p(s,d))\leqslant J_p$；(4)包丢失率约束：$lost(p(s,d))\leqslant L_p$；(5)费用约束：在所有满足式(1)~(4)条件的路径中 $cost(p(s,d))$最小。

**定义 2** $N=(S,T;F)$是一个受约束的原型 Petri 网，$\forall t\in T$，$|{}^{\bullet}t|=1$。$s$ 的托肯标签定义为：$TokenFlag(s)$，若，$t\in T$，$\exists M:M[t>M'$，$TokenFlag(s')=TokenFlag(s)\circ s$，其中 $s\in {}^{\bullet}t$，$s'\in t^{\bullet}$，“$\circ$”是一种连接运算。

初始标识 $M_0$ 下，$\forall s\in S$，$TokenFlag(s)=\phi$。

给定一个网络拓扑结构 $N^*=(V,E)$，其中 $V$ 是结点的集合，$E$ 是单向链路的集合。$s\in V$ 为源结点，$d\in V-\{s\}$ 是目的结点。显然一个网络拓扑结可采用受约束的原型 Petri 网模拟，模型转换范例如图 2 所示。

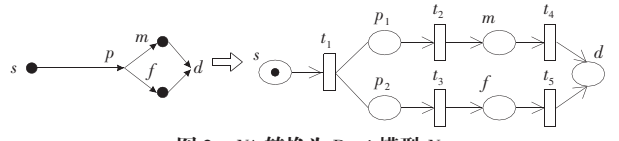



**图 2** $N^*$ **转换为** Petri **模型** $N$

令 $M_0(s)=1$，存在变迁序列 $\sigma_1=t_1t_2t_4$，$\sigma_2=t_1t_3t_5$的发生，每一个变迁序列的发生都将使库所 $d$ 含有一个带不同标签的托肯，最终 $TokenFlag(d)=s\circ p_1\circ m+s\circ p_2\circ f$。每一个标签对应 $N^*$ 中从结点 $s$ 到结点 $d$ 的一条路径。若充分考虑 QoS 所规定的性能度量参数，则可在这些路径中做出最优选择。

以上针对的是单向链路，对于实际网络中的双向链路，$s$ 到 $d$ 的最优路径，也意味着 $d$ 到 $s$ 的最优路径。

每个源服务器更关注自身所能提供的共享文件，具有简单的源服务器路由表。

```
(1)typedef struct{
    char Neighbor-NodeID;//存放与之相连的兄弟结点 NodeID+Ip+
        端口号
  } ServerRoutingTable;
```

每个代理服务器比源服务器更关注路由效率。具有复杂的代理服务器路由表。

(2)*typedef struct*{

char *Son-NodeID*;//存放与之连接的自治系统中的 *NodeID*+Ip+端口号

char *Destination-NodeID*;//目的地址的 *NodeID*+Ip+端口号

char *Best-Route*;//对应每一个目的地址的一个最佳路由，由所经过结点的 *NodeID*+*ip*+端口号组成

char *SecondBest-Route*;//相应于最佳路由的次佳路由，作为最佳路由的储备。同样由路径所经过的 *NodeID*+Ip+端口号组成

} *ProxyRroutingTable*;

定义源服务器和代理服务器共用的数据结构:

*typedef struct*{

*Element NodeID*;

*Element ProxyRroutingTable*;

*Element ServerRoutingTable*;

} *ServerNode*;

路由算法(Proxy-Ra)及结点的加入算法(InsertProxy)详细描述如下:

*typedef struct*{

*float TTL*;//该指令的生存周期

char *TokenFlag*;//该指令所带的标签

char *Parameter*;//结点及链路参数

char *Role*;//表明该信息是作为加入信息或路由信息

} *Message*;

*Message TransMessage*(*ServerNode Q*,*ServerNode R*,*Message Info*)//*Q* 收到 *R* 转发的信息

{ IF(*Info.TTL* is *invalid*)*Discard*(*Info*);//若 *TTL* 无效,丢弃该信息

ELSE IF(*Q* is *in Info.TokenFlag*)*Discard*(*Info*);//若已贴有自身的标签,抛弃该信息

ELSE { *Info.TokenFlag*=*Info.TokenFlag*∘*Q*;//在该信息标记上贴上自己的标签

*Info.Parameter*=*Info.Parameter*∪($n\_d(Q)$,$n\_c(Q)$,$n\_j(Q)$,$n\_l(Q)$,$l\_d(R,Q)$,$l\_b(R,Q)$,$l\_c(R,Q)$,$l\_j(R,Q)$);//附上结点 *Q* 及(*R*,*Q*)的度量指标

IF(*Info.Role*==*X* is *request Join*)*Send*(*Q*,*X*,*Info*);//返回信息给申请加入结点 *X*

IF (*Info.Role*==*search a Route* from *X* to *Y*)//需路由一条从 *X* 到 *Y* 路径

{IF($delay(p(X,Q))\leqslant D_p \wedge bandwidth(p(X,Q)\geqslant W_p \wedge jitter(p(X,Q))\leqslant J_p \wedge lost(p(X,Q))\leqslant L_p$)

IF(*Q.NodeID*==*X.NodeID*){*Info.TokenFlag*=*min*(*cost*(*p*(*X*,*Y*)))+*Secondmin*(*cost*(*p*(*X*,*Y*)));//若最佳路由有多条,则选取途径结点数最少的;若次佳路由有多条,则选取与最佳路由相异结点最多的路径 *Send*(*Y*,*X*,*Info*);}//将选取的路由返回结点 *S*

ELSE *Discard*(*Info*);}

IF(*Q.ServerRoutingTable*!=NULL)//*Q* 是源服务器,向所有与自己相连的结点(除 *R* 外)转发信息

{ For Every $Q_i$=*Q.ServerRoutingTable.Neighbor-NodeID-R*,in *Parallel* DO *TransMessage*($Q_i$,*Q*,*Info*);

Until(*Info.TTL* is *invalid*);}

IF(*Q.ProxyRroutingTable*!=NULL)//*Q* 是代理服务器,向所有与自己相连的结点(除 *R* 外)转发信息

{ For Every $Q_i$=*Q.ProxyRroutingTable.Destination-NodeID-R*,in *Parallel* DO *TransMessage*($Q_i$,*Q*,*Info*);

Until(*Info.TTL* is *invalid*);}

}

**算法 1** 代理服务器路由算法 *Proxy-Ra*( )

输入:代理服务器 *S*;

输出:*S* 与目的结点 *D* 的最佳路由及次佳路由。

*ServerNode Proxy-Ra*(*ServerNode S*,*ServerNode D*) //寻找从 *S* 到 *D* 的路由

{ *Message RouteMessage*;

*RouteMessage.Role*=search *a Route* from *S* to *D*;

For Every $B_i$=*B.ProxyRroutingTable.Destination-NodeID*,in *Parallel* DO *TransMessage*(*Bi*,*S*,*RouteMessage*);

Until(*RouteMessage.TTL* is *invalid*);

Receive(*S*,*D*,*RouteMessage*);//*S* 收到 *D* 回复而来的路由信息

Output(*RouteMessage.TokenFlag*);//输出 *S* 到 *D* 的最佳及次佳路由

}

**算法 2** 代理服务器加入算法 *InsertProxy*( )

输入:新加入的代理服务器 *P*;

输出:代理服务器 *P* 的资源表及路由表的建立。

*ServerNode InsertProxy*(*ServerNode P*) //*P* 申请加入代理层

{ *Message JoinMessage*;

*JoinMessage.Role*=*P* is request *Join*;

Broadcast(*P*,*JoinMessage*); //*P* 向源流层和代理层广播一条“加入”信息

For Every $A_i$ *receive the JoinMessage*,in *Parallel* DO

*TransMessage*($A_i$,*P*,*JoinMessage*);

Until(TTL is in valid);

*Receive* (*P*,*Q*,*JoinMessage*);//*P* 收到 *Q* 回复而来的关于加入信息的情况

$V'=V'\cup Q$;

For Every $x\in V'$,in *Parallel* DO

*P.ProxyRroutingTable.Destination-NodeID*=*x*; //*P* 根据所有收到的回复结点的情况构造自身的路由表

*P.ProxyRroutingTable.Best-Route*=min(*cost*(*p*(*P*,*x*)));

*P.ProxyRroutingTable.SecondBest-Route*=*Secondmin*(*cost*(*p*(*P*,*x*)));

*x.ProxyRroutingTable.Destination-NodeID*=*P*;//*x* 将 *P* 加入自己的目的结点,并构造相应路由

*x.ProxyRroutingTable.Best-Route*=min(*cost*(*p*(*P*,*x*)));

*x.ProxyRroutingTable.SecondBest-Route*=*Secondmin*(*cost*(*p*(*P*,*x*)));

Until(*traversal the element of V'*);

}

算法 2 的时间复杂度分析同算法 1,以下分两部分阐述对算法 1 的分析。本文所指的路径是两结点间没有重复结点和边出现的通路;如果问题的规模为 *M*,需求解网络中结点 *S* 与 *D* 的一次最佳路由:

(1)首先,存储 *M* 个结点的邻接矩阵 $\boldsymbol{C}=[C_{SD}]_{M\times M}$,定义 *C* 与 *C* 间的乘法运算得到 $\boldsymbol{C}^2=[C_{SD}^{(2)}]_{M\times M}$,其中 $C_{SD}^{(2)}=\sum_{k=1}^{M}C_{Sk}\cdot C_{kD}$。元素 $C_{SD}^{(2)}$表示 *S* 到 *D* 的长度为 2 的所有路径。同理,可得 $\boldsymbol{C}^{\gamma}=[C_{SD}^{(r)}]_{M\times M}$,$r=3,\cdots,M-1$,其中,$C_{SD}^{(r)}=\sum_{k=1}^{M}C_{Sk}\cdot C_{SD}^{(r-1)}$,$C_{SD}^{(r)}$是 *S* 到 *D* 间长度为 *r* 的所有路径。由矩阵 $C,C^2,\cdots,C^{M-1}$可得到 *S* 到 *D* 间的所有路径为:$\sum_{r=1}^{M-1}C_{SD}^{(r)}$,($C_{SD}^{1}=C_{SD}$)。

结点平均出度为 *OD*,结点 *S* 与 *D* 的一次最佳路由选取所需的托肯分裂次数 *Split* 为:$(OD)^{Split-1}=\sum_{r=1}^{M-1}C_{SD}^{r}\Rightarrow Split=1+\log_{OD}^{\sum_{r=1}^{M-1}C_{SD}^{r}}$。

不失一般性,可认为结点 $S$ 将托肯沿着深度为 $Split$ 的完全 $OD$ 叉树做广播。共有 $\sum_{j=1}^{Split-1} OD^j$ 条边。其中任两个结点间托肯资源的传输所需的平均时间为 $t=\frac{\widetilde{D}}{3\times10^8\ \text{m/s}}$，$\widetilde{D}$是服务器结点间的平均距离,$3\times10^8$ m/s 是通讯的传播速度。则所有托肯传递所需的时间 $t_1=t\cdot\sum_{j=1}^{Split-1} OD^j$。

(2)其次,$M-1$ 个结点共接收到 $\sum_{i=1}^{Split} OD^{i-1}$ 个托肯,每一个结点对每一个托肯的处理时间为 4 个单位时间(假设每个约束条件的处理占用一个单位时间)，则处理所有托肯信息所需的时间为 $t_2=4\cdot\sum_{i=1}^{Split} OD^{i-1}$。

因此,总的算法的时间复杂度为:$O(t_1+t_2)=O(\frac{\widetilde{D}}{3\times10^8\ \text{m/s}}\cdot\sum_{j=1}^{Split-1} OD^j+4\cdot\sum_{i=1}^{Split} OD^{i-1})=O(OD^{Split})=O(OD^{1+\log_{od}^{\sum_{i=1}^{k-1} c_{ai}}})=O(M^2)$。

每个代理服务器遇到有服务器结点的退出和加入,都更新一次自己路由表中的最佳和次佳路由;另外,代理服务器在一定的周期内也应主动执行一次路由更新过程。以保证所存储的最佳路由及次佳路由总是当前网络的两个结点间最优的路径。

### 3.2 性能分析

(1)服务器结点的离开

已假设每个服务器结点都性能稳定且不存在非正常掉线。结点的离线都认为是正常离开:

①若源服务器需要离开网络,则通知其路由表的邻居结点中的代理服务器结点，由它们发出一个源服务器离线通知(*Server Logoff*)，所有接到该通知的代理服务器查看自身的路由表是否含有该源服务器地址,若有,则做路由更新。

②若代理服务器需要离开网络,则首先在自身路由表中寻找离自己最近的且允许接纳新的子结点的代理服务器,通知自己的子结点更改父结点,给自身路由表中的所有目的结点发送代理服务器离线通知(*Proxy Logoff*),所有接到该通知的代理服务器查看自身的路由表中是否含有该代理服务器地址,若有,则做路由更新。

(2)表项的维护及更新代价

①若现有一代理服务器结点 $X$,则 $X$ 的路由表存储容量满足:$Capability_X=150\ \text{bit}\cdot x+2(M-1)^2\cdot150\ \text{bit}$。

其中 $M=\frac{TTL\cdot3\times10^8\ \text{m/s}}{\widetilde{D}}$,$M$ 是与 $X$ 所连接的目的结点的个数;$\widetilde{D}$为服务器结点间的平均距离;考虑往返,取 $TTL=t_{deadline}/2^2$。$X$ 至多管理 $x$ 个自治系统。

$t_{deadline}$ 的取值由以下公式获得:

$$(\frac{TTL\cdot3\times10^8\ \text{m/s}}{\widetilde{D}})^3\cdot\frac{1}{2\cdot10^9}\ \text{s/次}+\frac{4\cdot10^7\ \text{m}}{3\cdot10^8\ \text{m/s}}=TTL\qquad(1)$$

其中 $3\times10^8$ m/s 是通讯的传输速度,$\frac{1}{2}\cdot10^9$ s/次是主流服务器的 CPU 能力。$4\cdot10^7$ m 是赤道周长(考虑一次请求应足以传递整个 Internet 网)。

假设 $x=100$,$\widetilde{D}=500$ Km。由公式(1),$TTL=0.1$ s,$t_{deadline}=0.4$ s,$Capability_X=1$ Mbit。

②一次路由更新网络中桥所需带宽:$W=(((M-1)\cdot150\ \text{bit}+(M-1)\cdot8\cdot5\ \text{bit})\cdot OD))/t_{i+1}-t_i$。$M$ 为问题的规模,$OD$ 为代理服务器的平均出度，最坏情况下每一段含 8 个 QoS 的性能度量参数,每个参数 5 bit。($t_i$,$t_{i+1}$)为两次托肯分裂之间的时差,即结点间托肯资源的传输时间,$t_{i+1}-t_i=\frac{\widetilde{D}}{3\times10^8\ \text{m/s}}$。

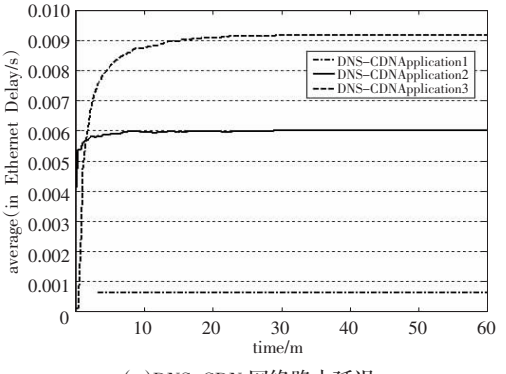
若 $x$,$\widetilde{D}$,$t_{deadline}$ 的取值如上,另取 $OD=30$,得 $W=200$ Mbit/s。

## 4 实验仿真

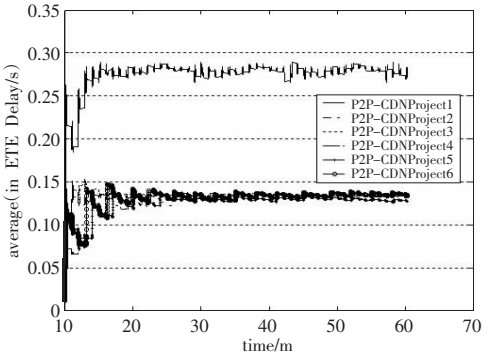
以 OPNET Modeler10 为仿真平台,开发了两个原型系统:传统基于重定向服务器(DNS)的 CDN 网络(DNS_CDN)及以图 1 为网络架构的 CDN 与 P2P 结合的 P2P_CDN。以观测不同负载,不同网络结点数的条件下路由算法的平均网络延时。在实验中,网络结点数最多时达到 1 000。

为了评价两种网络结构下的路由效率,让网络中结点逐渐增多,每个结点的路由请求数逐渐增大。图 3 中,$X$ 轴表示实验模拟时间,$Y$ 轴表示平均路由延时。图 3(a)及图 3(b)分别代表 DNS_CDN 网络及 P2P_CDN 网络的实验数据。各曲线所代表参数如表 1(a),(b)所示。

从图 3(a)可以看出,随着网络结点数及每个结点路由请求的增多,重定向服务器逐渐成为新的网络瓶颈,造成所有网络结点平均路由延迟快速从 0.001 2 s 增加到 0.09 s。图 3(b)可以看出,除非在极端情况下,给予结点数 1 000 的网络以较大




(a)DNS_CDN 网络路由延迟




(b)P2P_CDN 网络路由延迟

**图 3　不同网络拓扑下平均路由延时**

**表 1 各实验曲线参数**

(a)DNS_CDN 网络

| 曲线 | 网络中结点数 | 每个结点单位时间内发起的路由请求 |
|---|---|---|
| DNS_CDNApplication1 | 500 | 以 Uniform_int 函数每秒随机发送 1~5 个 |
| DNS_CDNApplication2 | 1 000 | 以 Uniform_int 函数每秒随机发送 1~5 个 |
| DNS_CDNApplication3 | 1 000 | 以 Uniform_int 函数每秒随机发送 10~20 个 |

(b)P2P_CDN 网络

| | | |
|---|---|---|
| P2P_CDNProject1 | 1 000 | 以固定速率每秒发送 50 个 |
| P2P_CDNProject2 | 1 000 | 以 Uniform_int 函数每秒随机发送 15~20 个 |
| P2P_CDNProject3 | 1 000 | 以 Uniform_int 函数每秒随机发送 10~15 个 |
| P2P_CDNProject4 | 1 000 | 以 Uniform_int 函数每秒随机发送 1~5 个 |
| P2P_CDNProject5 | 500 | 以 Uniform_int 函数每秒随机发送 1~5 个 |
| P2P_CDNProject6 | 300 | 以 Uniform_int 函数每秒随机发送 1~5 个 |

固定速率的路由请求,其延迟达到 0.27 s。其它不同结点数,不同随机函数的路由搜索延迟都相对较稳定,平均值取 0.14 s。从图中可以看出,当网络中结点数较少时,DNS_CDN 网络的路由效率要优于 P2P_CDN 网络。但随着结点数或每个结点的路由请求的增加,前一种网络的路由效率显著下降,而后一种网络则呈现良好的稳定性。

实验中还获得了与 3.2 节理论分析值基本吻合的实验数据,限于篇幅,本文在此不再罗列。

## 5 结论和进一步研究

本文在 Petri 网,蚁群算法的基础上针对 CDN 网络对 QoS 的要求提出了一种新的路由算法。与基本 Petri 网相比,路由过程增加了信息素功能。每次变迁实施时,托肯都会在变迁上留下信息素来影响后来托肯的路径选择,使蚁路最终逼近于全局最短的时间变迁序列。文章通过具体的算法解决了蚁群算法和 Petri 网在 CDN 应用的具体问题,在一定程度上解决了 CDN 路由对QoS 的要求。仿真结果证明采用该算法能够快速的找到全局最优路径,且具有较好的可扩展性和鲁棒性。但本文的一个重要假设是各个结点性能都稳定可靠,这就限制了本算法的使用范围,随后还需要针对动态随机 CDN 进行更深入的研究。

**致谢** 在此,向朱涛,陈勇,刘松,邓江等表示感谢,他们为本文提供了详尽的实验数据,为完善理论研究提供了有益的帮助。

## 参考文献:

[1] Stardust.com Inc.White paper-the ins and outs of content delivery networks[R/OL].(2001).http://www.stardust.com/cdnweek/whitepapers/cdnspring01.San Francisco:Stanford University.

[2] IETF Internet-Draft.Known CDN request-routing mechanisms[EB/OL].(2001-05-18)[2007-09-10].http://www.tools.ietf.org/html/draft-cain-cdnp-known-request-routing-00.

[3] Gary M R,Johnson D S.Computers and intractability:a guide to the theory of NP-completeness[M].San Francisco:W.H.Freeman and Company,1979.

[4] Xiao X P.Providing quality of service in the internet[D].Michigan:Michigan State University,2000.

[5] Dorigo M,Gambardella L M.Ant colonies for the traveling salesman problem[J].Bio-system,1997,43(2):73-81.

[6] Dorigo M,Caro G D,Stutzle T.Ant algorithms[J].Future Generation Computer System,2000,16(1):5-7.

[7] Michlmayr E,Pany A,Kappel G.Using taxonomies for content-based routing with ants[J].Computer Networks,2007,51(16):4514-4528.

[8] van der Aalst W M P,van Hee K.Workflow management models,methods and systems[M].Cambridge:MIT Press,2002.

[9] Reisig W.Petri nets:an introduction[M].Berlin,Heidelberg:Springer-Verlag,1985:17-135.

[10] 袁崇义.Petri 网原理与应用[M].北京:电子工业出版社,2005.

(上接 27 页)

## 5 结论

本文提出的基于层次聚类的弱小目标检测方法,利用恒星、目标和噪声的不同运动规律来区分恒星,因此对星点灰度变化不敏感,抗噪声干扰能力强,利用了目标的连续性和一致性,可以准确区分目标和噪声。实验证明,该方法只需要三帧序列图像就可以有效检测移动背景空间图像中的恒星和弱小目标,具有较强地实用性,而最常见的基于背景模型的检测方法无法区分目标和恒星。

## 参考文献:

[1] Irani M,Anandan P.A unified approach to moving object detection in 2D and 3D scenes[J].IEEE Transaction on Pattern Analysis and Machine Intelligence,1998,20(6):577-589.

[2] Jones R,Ristic B,Redding N J.Moving target indication and tracking from moving sensors[J].Digital Image Computing:Techniques and Applications,2005,6(8):46-54.

[3] 赖作镁,王敬儒,张启衡.背景运动补偿和假设检验的目标检测算法[J].光学精密工程,2007,15(1):112-116.

[4] 程德杰,李晓峰,李在铭.基于场景运动分析的弱小目标形态学检测方法[J].电子测量与仪器学报,2006,20(3):1-5.

[5] 张宇,吴宏刚,陈跃斌,等.采用形态神经网络背景自适应预测的图像弱小目标检测[J].计算机应用研究,2007,24(3):289-291.

[6] 卓宁,孙华燕,张海江.红外图像中弱小目标检测算法概述[J].光学仪器,2005,27(4):83-86.

[7] 汪国有,陈振学,李乔亮.复杂背景下红外弱小目标检测的算法研究综述[J].红外技术,2006,28(5):287-292.

[8] Ristic B,Arulampalam S,Gordon N.Beyond the Kalman filter:particle filters for tracking applications[M].Boston:Artech House,2004.

[9] 张长城,杨德贵,王宏强.红外图像中弱小目标检测前跟踪算法研究综述[J].激光与红外,2007,37(2):104-107.

[10] Duda R O,Hart P E,Stork D G.模式分类[M].2 版.北京:机械工业出版社,2003.

[11] Hastie T,Tibshirani R,Friedman J.统计学习基础——数据挖掘、推理与预测[M].北京:电子工业出版社,2004.