可信可控网络中的一致性视图构建机制

刘泽民

(西安交通大学电子与信息工程学院 西安 710049) (攀枝花学院数学与计算机学院 攀枝花 617000)

摘 要 可信可控网络利用多个控制节点对AS进行联合控制,容易造成多个控制节点在网络控制过程中持有的AS视图不一致。针对该问题,在可信可控网络模型的基础上提出了基于选举算法的AS内一致性视图构建机制,该机制首先基于选举算法选举出主控制节点,然后主控制节点根据AS内各个控制节点的负载将视图构建任务分配给负载最低的控制节点负责构建视图,并利用主控制节点的时间对生成的视图的版本进行界定,从而避免了多个控制节点独自构建视图造成的视图混乱问题。另外,仿真实验结果表明,提出的一致性视图构建机制具有良好的性能。

关键词 可信可控网络模型,选举算法,一致性视图

中图法分类号 TP301 **文献标识码** A

Consistent View Construction Mechanism in Trustworthy and Controllable Network

LIU Ze-min

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

(School of Mathematics and Computer Science, Panzhihua University, Panzhihua 617000, China)

Abstract Multi control nodes are used to control an AS coordinately in Trustworthy and Controllable Network. The views of different control nodes may be inconsistent. To solve this problem, we proposed a consistent view construction mechanism, in which a selection algorithm is used to generate a primary control node first, and then the primary control node is responsible for organizing other control nodes to build consistent view for all view requests. This mechanism avoids the problem that the views of different control nodes are inconsistent. Besides, the simulation experiment verifies that the mechanism proposed in this paper has better performance than prior view construction method in both time cost and communication overheads.

Keywords Trustworthy and controllable network model, Selection algorithm, Consistent view

1 引言

可信可控网络将网络控制逻辑的计算集中到控制节点(Control Node,CN)上,以便于对网络建立统一的网络控制层面。但是单一控制节点容易导致性能瓶颈和单点故障问题,为了提高可信可控网络的可扩展性,在AS内采用多控制节点对网络进行协同控制已成为相关研究者的共识[1,2]。

虽然在可信可控网络中多个控制节点对网络进行协同控制的方法能够有效地提高可信可控网络的可扩展性和鲁棒性,但是它也给可信可控网络带来了新的视图一致性的问题:在可信可控网络中,每个控制节点在控制域内维护一个控制信息数据库来保存本控制域的控制信息,而在每个控制节点上运行着多个网络控制机制,由于网络延迟以及网络震荡性等原因,如果多个控制节点的多个控制机制都需要自己构建AS的网络视图,很容易造成视图信息的不一致,从而导致不同网络控制机制对网络的控制策略产生矛盾,影响网络的正常运行。

为了便于说明多个控制节点网络视图可能不一致的问题,我们给出了一个示例:如图1所示,控制节点CN1和CN3分别在t_1和t_2时刻开始构建各自的网络视图,分别向除自身以外的所有控制节点发送其他控制域的视图请求。而在CN1和CN3构建网络视图的过程中,CN4负责的控制域的网络状态在t_3时刻发生了变化,由于CN1与CN4的网络延迟较长,使得CN4对CN1返回的网络视图发生在t_3以后。而由于CN3与CN4的网络延迟较短,使得CN3在t_3时刻之前就返回了其局部视图,这就会造成CN1在t_5得到的视图与CN3在t_4时刻得到的视图不一致。而由于CN3与CN2间的通信延迟,造成CN3得到视图的时刻t_4晚于CN1得到视图的时刻t_5,这就造成生成时刻较晚的视图其准确性反而不如生成时刻较早的视图的高。

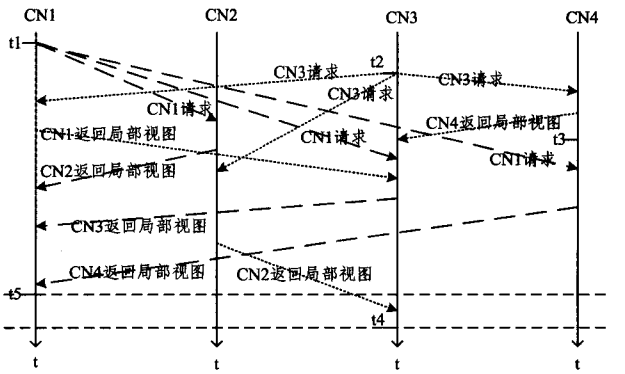

图1 控制节点分别构建网络视图示例

到稿日期:2012-09-05 返修日期:2012-12-18 本文受国家自然科学基金(61071217)资助。

刘泽民(1963—),男,硕士,副教授,主要研究方向为机器智能、计算机图形、图像处理,E-mail:liuzemin998@qq.com。

从上述示例可以看出，由每个控制节点单独生成视图会造成视图的一致性问题，主要表现在两个方面：

(1)控制节点的视图与网络真实状态不一致。由于分布式网络环境下传输时延和计算耗费时间等原因，很难构造与网络实际状态严格吻合的视图，网络视图与网络实际状态必然会存在一个时间差。为了解决这一问题，一般采用不断更新视图版本的方法来反映网络的实际状态。因此，这个问题可以归结为多个控制节点组成的分布式网络环境下，缺少全局时间对网络视图进行版本界定，造成网络视图版本前后不一致性问题。如图1给出的例子中，CN3在 t_4 时刻得到的视图不如CN1在 $t_5(t_4>t_5)$ 时刻得到的视图准确，就是新老视图的不一致性问题。

(2)多个控制节点的视图互相矛盾。由于不同控制节点独自构建视图，造成视图来源混杂，导致网络视图混乱，很难判断它们所构建的视图的正确性。如图1给出的例子中CN3得到的视图没有反映 t_3 时刻以后CN4对应的控制域内的网络状态变化，与CN3得到的视图矛盾。

因此，为了解决上述的视图版本前后不一致问题和不同控制节点视图不一致问题，本文给出了可信可控网络中的一致性视图构建机制，该机制主要由主控制节点选举算法和基于选举的一致性视图构建算法两部分组成。其主要思想是当网络控制机制提出一致性视图构建需求时，其所在的控制节点将任务提交给选举算法产生主控制节点；然后主控制节点根据控制节点的优先级将该任务分配给优先级最高的节点处理；负责处理的控制节点负责组织多个控制节点基于合作的方式构建一致性视图后，将最终的处理结果发送给主控制节点；主控制节点再将处理结果发送给请求方，保证多个控制节点的网络视图源一致，解决了多个控制节点的视图不一致问题，并且通过利用主控制节点的时间戳作为版本号的方式，避免了网络视图前后版本的不一致性。

2 AS内的主控制节点选举算法

基于选举算法的一致性视图构建机制主要通过为AS选出主控制节点的方法为多个控制节点的网络视图提供统一的版本界定方法和视图来源。在本节中，将根据可信可控网络的特点，为多个控制节点构建一个主控制节点选举算法。

选举算法是分布式系统中常用的算法，其从进程集合中选出一个特定的进程来对特定任务进行处理，以实现多个进程之间的协作。选举算法应用的领域很多，如群服务器、重复数据更新、负载均衡、应急恢复以及互斥等方面。根据不同网络的拓扑类型，人们提出了不同的分布式选举算法[3]。一般情况下选举过程可以分成两个阶段：第一，选择具有最高优先级的优胜者；第二，通知其他参与者谁是优胜者。两个阶段都需要在系统中发布进程的ID，因此通过进程之间的通信方式可以将选举算法分成两种：面向广播网的选举算法[4]和面向存储转发网的选举算法，其中面向存储转发网的选举算法，又有面向单向环的选举算法[5-7]、面向双环的选举算法[8]、面向完全图的选举算法[9]和面向弦环的选举算法[10]。

根据上述在分布式网络环境下的选举算法，针对可信可控网络中视图构建的一致性需求和多个控制节点的负载均衡，我们基于bully算法设计了可信可控网络中的主控制节点选举算法和基于合作的视图构建算法。

针对可信可控网络中AS内多个控制节点的一致性视图构建问题，我们基于传统的bully算法[9]设计了主控制节点选举算法。由于在可信可控网络中每个控制节点只有一个参与选举的进程，为了便于理解，在后文中统一将参与选举的对象确定为AS内的控制节点。

2.1 优先级定义

为了统一各个CN负载的评价标准，给出了一个以请求需要等待的时间作为衡量标准的方法，并选择响应当前请求所需要的时间最短的CN作为处理当前请求的节点。每个节点对当前请求响应时间的计算方法如下：

$$T_i = p * \lambda_i \tag{1}$$

式中，p 表示在节点 i 上排队的任务数，λ_i 表示节点 i 处理一个任务所花费的平均时间。利用 T_i 定义每个节点的优先级，如式(2)所示：

$$priority_i = \begin{cases} 0, & T_i = 0 \\ -T_i, & T_i \neq 0 \end{cases} \tag{2}$$

由式(2)所示，节点的响应时间越长，其优先级越低，当其响应时间为0时，表示该节点空闲，其节点优先级最高。

2.2 算法介绍

为了能响应用户提出的请求，将CN分成主控制节点(Primary Control Node，PCN)和从控制节点(Secondary Control Node，SCN)，PCN负责进行任务管理，SCN负责对用户的任务进行处理，处理某个用户请求的SCN成为当前任务的责任节点。在一个AS内的多个CN组成了一个服务组响应用户请求，并通过选举算法选出一个PCN。每个CN都有自己的一个公共组播地址和私有IP。

经典的bully算法[9]是一个可靠的选举算法。基于bully选举算法设计了我们的选举算法，过程如下：

CN进入选举状态后，向其他CN发送选举组播包，并启动定时器T1，等待AS中其它CN的响应包。当节点发现有更高优先级选举CN时，就自动设置状态为SCN状态。如果在T1时间内，没有收到更高优先级CN的选举包，则设置状态为PCN状态。

在SCN状态，CN启动一个定时器T2，如果在T2允许的时间内获得PCN发来的轮询包，就做出响应，并更新CN优先级队列，将PCN发送来的任务加入任务队列。如果在定时器T2允许之内没有获得PCN发来的轮询包，则进入选举状态。如果在T2允许的时间内收到其它SCN的选举包，则先比较负载，若自身的负载比较低，则进入选举状态。

进入PCN状态后，首先停止自己的任务并将任务移送至CN优先级队列中优先级高的CN上，然后开始监控整个AS中其他ACN的状态，监控过程为：启动定时器T3，并向所有SCN发送轮询包收集SCN的状态和任务信息。如果有新任务，就将任务分配给最高优先级的CN并更新SCN优先级队列，然后将形成的新的SCN队列和任务信息发送至SCN。如果在T3允许的时间内又有新的选举包到来，就发送回复，中止其选举。

为了更好地说明选举算法的过程，图2给出了主节点选举算法的伪代码，具体如下：

```
Selection-algorithm{
  State=SCN;//初始化当前节点状态
While(1){
  if(state==SCN){
    T2.val=0;//启动定时器 T2
    waitfor(t2);//等待 t2 时间
```

```
    If (have_received(PCN_ASK)){ //如果在 t2 内收到 PCN 轮
      询包,当前节点变为 SCN 状态
      revise_task_queue();
      revise_priqueue();
      send_ACK(PCN,LoadState);
    }else{
      if (have_received(SELECT)){ //如果在 t2 时间内收到选举包
        if (have_higher_priority(selectsender)){ //且具有更好的优
          先级,则进入选举状态 selection(IP);
        }
      }
      State=select;//如果在 t2 时间内没收到 PCN 轮询包,进入选
        举状态
      select(IP);
  }
}else{
  if (state==PCN){ //如果当前节点是 PCN,则发送 PCN_ASK 轮
    询包询问其他节点的状态,并将任务和优先级队列发送给 SCN
    T3. val=0;//启动定时器 T3
    wait(t3);
    send(PCN_ASK,allSCN);
    send(priqueue,allSCN);
    send(task,highest_priority_CN);
    if(have_received(SELECT)){
      send (SELECT_STOP,IP)//如果在 t3 时间内收到选举包,则
          发送选举终止包制止选举
    }
  }
  }
}
```

(a)节点状态维护算法

```
select(IP){ // 选举函数
  send(SELECT,allCN,IP);//向其他 CN 发送选举组播包
  T1. val=0 //启动计时器 T1
  wait(t1);
  if (have_receive(higher_priority_CN)){//当节点发现有更高优先
    级选举 CN 时,就自动设置状态为 SCN 状态。
  state=SCN;
  }else{
  state=PCN;//如果没有收到更高优先级节点的选举包,则自己成
  为 PCN
    send(task,highest_prirority_CN_in_priqueue)
  }
}
```

(b) 主节点选举算法

图 2 PCN 选举算法

根据上述的选举过程,图 3 给出了相应的状态转移图。节点的状态可以分成 3 种:选举态、PCN 态和 SCN 态。每个 CN 都维持一个守护进程,在正常情况下一个 AS 内有一个 PCN,当节点刚加入时默认设置为 SCN 状态。

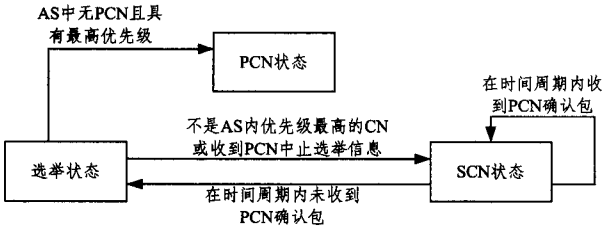

图 3 AS 中 CN 的状态转移图

3 基于选举的一致性视图构建算法

3.1 算法介绍

第 2 节中,给出了 AS 内主控制节点的选举算法。在本节中基于选举算法的选举结果,并根据网络视图的一致性需求和控制节点之间的负载均衡的需求,提出了一致性视图构建机制来对用户(网络控制机制)提交的网络视图构建任务进行统一处理。其主要思想是需要构建 AS 内一致性视图的用户统一向主控制节点提交一致性视图构建请求;主控制节点将任务分配给从控制节点进行处理,同时设定一个任务等待时间,并在视图构建任务完成前将提交一致性视图构建请求的用户放在等待队列中,当一致性视图构建任务处理完成后,一次性将一致性视图交付给等待队列中的用户;而从控制节点则负责根据主控制节点分配的任务构建该控制域的局部视图,并由其中一个从控制节点对这些局部视图进行综合后提交给主控制节点。为了更清楚地说明处理过程,给出了一致性视图构建算法的步骤,具体如下:

1)由于 AS 内的每个控制节点都知道本 AS 内的主控制节点的地址,运行在某一控制节点上的网络控制机制产生一致性视图构建需求时,向 AS 的主控制节点 PCN 提交网络一致性视图构建请求 ViewRequest1,并设定一个时间 QT,如果在 QT 时间内没有收到 PCN 的一致性视图构建结果,则构建视图失败。

2)AS 的主控制节点 PCN 维持一个视图请求等待队列 VQueue 和一个视图构建计时器 PViewtimer,当 PCN 接收到 ViewRequest1 后,检查 WQueue 中是否存在正在等待的其他节点的视图处理请求,如果没有,则生成一个视图构建任务 ViewTask1,为该任务生成一个唯一的任务标识 TaskID,并将 ViewRequest1 添加到 VQueue 中,同时将 PViewtimer 清零;如果有,则仅将 ViewRequest1 添加到 VQueue 中。生成网络一致性视图构建任务 ViewTask1 后,PCN 查看其维护的控制节点优先级队列,并从优先级队列中选出具有最高优先级的从控制节点 SCN1,然后将网络一致性视图构建任务 ViewTask1 和任务标识 TaskID 分配给从控制节点 SCN1,并设定 PViewtimer 的等待时间为 PT 以等待 SCN1 返回处理结果,如果 PT 时间内没有获得 SCN1 的一致性视图构建结果,则构建视图失败。其中由于从控制节点 SCN1 负责对 ViewTask1 任务的处理,我们称从控制节点 SCN1 为责任从处理节点(Response Secondary Control Node,RSCN)。

3)RSCN 收到主控制节点 PCN 分配的网络一致性视图构建任务 ViewTask1 和任务标识 TaskID 后,向其他从控制节点 SCN 发送网络一致性视图合作构建请求 CorRequest2 和任务标识 TaskID,并设定等待时间 RT 等待其他从控制节点 SCN 返回处理结果,同时 RSCN 根据 ViewTask1 的需求构建本控制域的局部视图,如果其中存在一个协助构建视图的从控制节点没有在 RT 时间内完成局部视图构建,则一致性视图构建任务失败;其中除了 RSCN 以外的参与到任务 ViewTask1 处理的从控制节点的任务是协助 RSCN 构建 AS 的一致性视图,因此,我们将这些从控制节点称为协助从控制节点(Joint Secondary Control Node,JSCN)。

4) JSCN 接受到处理任务 CorRequest2 和任务标识 TaskID 后,根据 ViewTask1 任务的需求构建相应控制域的局部视图,并将结果和任务标识 TaskID 反馈给 RSCN。

5)RSCN 收到各个 JSCN 的处理结果和任务标识 TaskID 后，检查任务标识 TaskID 是否与当前处理的任务的标识一致，如果不一致则丢弃，如果一致则继续执行；如果每个 JSCN 的局部构建视图成功，则综合各个控制域的局部网络视图生成 AS 的一致性视图，并将该视图反馈给主控制节点；如果存在一个 JSCN 构建视图失败，则向 PCN 反馈构建一致性视图失败的信息。

6)主控制节点收到处理结果和任务标识 TaskID 后，如果任务标识 TaskID 与当前处理任务一致，则将一致性视图的构建结果反馈给提出请求的网络控制机制，否则将收到的结果丢弃，并继续等待。

图 4 给出了 AS 一致性视图构建过程示意图，图中的 AS 划分为 5 个控制域，对应有 5 个控制节点，用户（网络控制机制）提出构建视图的请求后，其构建过程按照上面的步骤在图中已被标出。

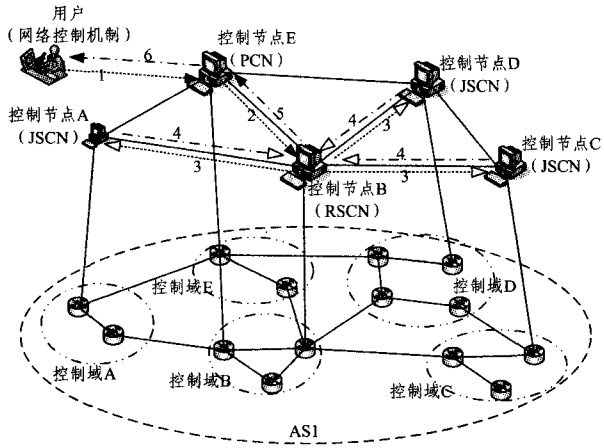

图 4　AS 一致性视图构建示意图

另外值得说明的是，由于在分布式网络环境下网络延迟以及视图计算的时间耗费等原因，严格实时的网络一致性视图是很难构建的，网络视图与网络实际状态必然会存在一个时间差，因此一般采用不断更新视图版本的方法来反映网络的实际状态。为了防止网络一致性视图由于时间延迟原因造成的网络控制的混乱，我们在可信可控网络中都以 AS 主控制节点的时间为准，对主控制节点生成的每个视图以一个时间戳为标准生成视图版本版本号，并以五元组（PCN，View，Vision，RequestTime，ViewGenerateTime）对视图进行标识，以避免网络新老视图的不一致问题。该五元组表示主控制节点 PCN 在 RequestTime 时刻接到用户请求后，在 ViewGenerateTime时刻构建成功的版本 Vision 的网络视图 View，其中的 RequestTime 和 ViewGenerateTime 都是以 PCN 的时间为标准。该五元组利用 PCN 的局部时间完成了对网络视图的统一的版本界定，从而能够保证各个控制节点之间视图不会产生矛盾。

由于 PCN 等待 RSCN 返回一致性视图以及 RSCN 等待 JSCN 返回局部性视图时都设定了固定的等待时限，等待超时就将视图构建任务判为失败，因此容易发生构建新版本视图时收到旧版本的网络视图或者旧版本的局部视图的情况，造成网络视图的混乱。为了解决此问题，我们为每个视图构建任务生成一个唯一性的任务标识 TaskID，在 PCN 收到 RSCN 返回的一致性视图和 RSCN 收到 JSCN 返回的局部视图时都要检查收到的结果的 TaskID 是不是与当前处理的任务的 TaskID 一致，如果不一致则将收到的结果丢弃，只有收到的结果的 TaskID 与当前正在处理的任务的 TaskID 一致时才能接受。

3.2　算法分析

在第 1 节的需求分析中，给出了网络一致性视图构建需要解决的两个问题：新老网络视图一致性问题和不同控制节点的视图不一致性问题。针对这两个问题，将对基于选举的一致性视图构建算法构建的视图进行一致性分析。

在上节给出的一致性视图构建算法中，每个 AS 的主控制节点 PCN 是该 AS 视图的唯一源头，并且以 PCN 的局部时间作为视图版本界定的标准，解决了视图版本前后矛盾和多个控制节点不一致的问题。利用基于选举的一致性视图构建算法生成的视图具有如下两个定理。

定理 1　将用户请求按照 PCN 的局部时间排序后，请求时间较晚的用户获得的视图（PCN，$View_B$，$Vision_B$，t_B，vt_B）的版本一定不低于请求时间较早的用户获得的视图（PCN，$View_A$，$Vision_A$，t_A，vt_A），即存在两个视图（PCN，$View_A$，$Vision_A$，t_A，vt_A）和（PCN，$View_B$，$Vision_B$，t_B，vt_B），如果 $t_B \geqslant t_A$，则 $vt_B \geqslant vt_A$，即 $Vision_B \geqslant Vision_A$。其中 t_A 和 t_B 是 PCN 接受任务处理请求的时刻。

证明：为了比较请求时间较晚的视图和请求时间较早的视图的版本高低，可以分两种情况进行讨论：第一种情况是较晚的视图请求到达时，较早的视图请求正在被处理；第二种情况是较晚的视图请求到达时，较早的视图请求已经被处理完成。具体如下：

第一种情况：如图 5 所示，PCN 分别在 t_A 和 t_B 时刻接受了用户 A 和用户 B 的请求，由于用户 A 的请求 QA 比用户 B 的请求 QB 接受的时间早，因此 PCN 接受 QA 时创建一个一致性视图构建任务，并设定时限 PT 等待视图构建；而当 PCN 接受 QB 时，一致性视图构建任务正在进行，视图等待队列 Queue 不为空，因此 PCN 仅将 QB 加入到等待队列中，以创建新的视图构建任务。当视图构建任务完成后，PCN 同时将视图发送给用户 A 和用户 B，故 $vt_B = vt_A$。

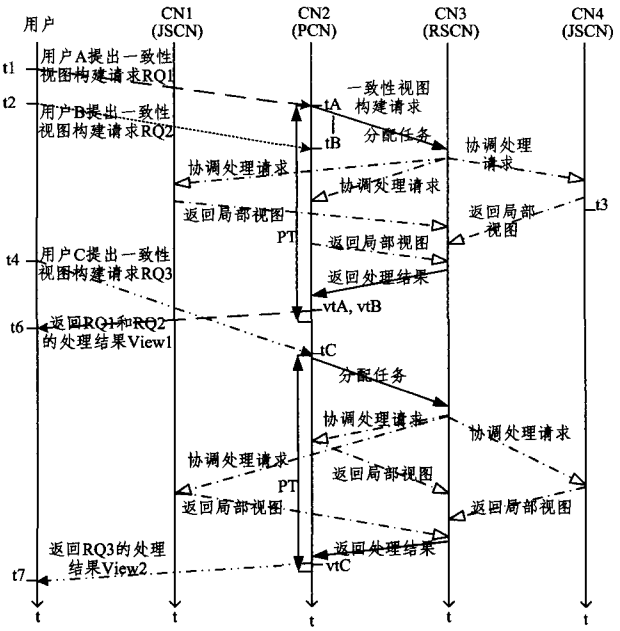

图 5　基于选举的一致性视图算法为多个用户构建网络视图示例

第二种情况：如图 5 所示，PCN 分别在 t_A 和 t_C 时刻接受了用户 A 和用户 C 的请求，由于 PCN 接受用户 C 的请求 QC 时，用户 A 的请求 QA 已经处理完成，很显然，$vt_C > vt_A$。

所以，综合上述两种情况，定理得证。

证毕。

定理 2 将用户获得的视图按照 PCN 的局部时间排序后，生成时间较晚的视图的版本高于生成时间较早的视图的版本，并且版本高的视图比版本低的视图准确。即两个视图（PCN，$View_A$，$Vision_A$，t_A，vt_A）和（PCN，$View_B$，$Vision_B$，t_B，vt_B）中，如果 $vt_B \geqslant vt_A$，则 $Vision_B \geqslant Vision_A$，并且 $Vision_A$ 和 $Vision_B$ 对应的网络真实状态（$RealState_A$，t_{nA}）和（$RealState_B$，t_{nB}）满足 $t_{nB} \geqslant t_{nA}$。

证明：由于网络视图是由 PCN 统一构建的，并且视图的构建时刻都是按照 PCN 的时间为标准的，因此构建时间较晚的视图必然比构建视图较早的视图的版本高，即两个视图（PCN，$View_A$，$Vision_A$，t_A，vt_A）和（PCN，$View_B$，$Vision_B$，t_B，vt_B）中，如果 $vt_B \geqslant vt_A$，则 $Vision_B \geqslant Vision_A$。并且由于 $View_B$ 的构建时间比 $View_A$ 的构建时间晚，因此 $View_B$ 获得的信息必然更新，即 $Vision_A$ 和 $Vision_B$ 对应的网络真实状态（$RealState_A$，t_{nA}）和（$RealState_B$，t_{nB}）满足 $t_{nB} \geqslant t_{nA}$。

证毕。

从上面两个定理可以看出，在可信可控网络中利用基于选举的一致性视图构建算法为多个控制节点统一构建网络视图的方法保证了网络视图的一致性。

4 多粒度视图构建

在前文中，给出了在可信可控网络中 AS 内为多个控制节点构建一致性视图的算法。在可信可控网络中，给出了可信可控网络中控制信息描述模型[10]，以及对这些信息进行存储的控制信息数据库。在本节中我们将基于该算法和可信可控网络控制信息数据库为 AS 内多个控制节点提供网络设备粒度和网络协议粒度的一致性视图。

AS 内设备粒度的一致性视图构建：设备粒度的视图构建主要为用户提供以 AS 内网络设备的邻接关系为主体构建的网络拓扑视图。按照上文给出的一致性视图构建方法，首先当用户提出构建设备粒度视图的请求后，主控制节点接受该请求，并将构建任务分配给优先级最高的 RSCN 负责构建视图，RSCN 将协调构建任务分配给各个 JSCN 后，各个 JSCN 分别在其负责的控制域内构建设备粒度的局部视图，并返回给 RSCN，RSCN 根据各个 JSCN 返回的局部视图构建 AS 内设备粒度的一致性视图后，将一致性视图返回给 PCN，最后 PCN 将结果返回给发出请求的用户。

AS 内协议粒度的一致性视图构建：协议粒度的视图在网络拓扑视图的基础上进行细化，将运行在传输网络上的协议之间的通信关系在视图上反映处理。由于协议粒度的视图比设备粒度的视图能够更详细地反映网络状态以及各个协议之间的依赖关系，因此构建协议粒度的一致性视图对网络管理和网络控制更有帮助。在可信可控网络的每个控制节点上都对其负责的控制域内运行的协议状态信息进行采集，并存储在控制信息数据库中，使得构建 AS 内的协议粒度的一致性视图成为可能。AS 内协议粒度的一致性视图构建方法与设备粒度的视图构建方法一样，也是利用本节中给出的一致性视图构建方法进行构建。

5 仿真实验

在可信可控网络网络中，SCVCA 保证了多个控制节点获取 AS 内的一致性视图。然而在多个控制节点对 AS 进行联合控制的过程中，需要保证视图能够在较短的时间和较低的负载下构建。为了验证 SCVCA 在构建时间和构建负载方面的性能，在仿真实验环境下对 SCVCA 和每个控制节点单独构建的方法（Single Construction Method，SCM）在时间和负载两个方面进行了比较。为了更好地验证 SCVCA 的性能，本论文利用仿真工具 NS2 构建了仿真实验。为了检验 SCVCA 在不同网络规模下的性能，随机生成了 4 个 $m+n$ 不同规模的实验网络环境，其中 m 表示控制节点的个数，n 表示路由器的个数，网络规模分别为 3+8，5+26，8+57 和 12+105。

视图构建所需要的时间直接影响到网络控制的有效性。为了比较 SCVCA 与 SCM 两种视图构建算法的构建时间，在上述网络环境下给出了两种算法的构建时间比较，如图 6 所示。从图中可以看出 SCVCA 构建视图所需要的时间在各种网络规模下都优于 SCM，并且可以看出随着网络规模的增大和控制节点数量的增加，SCVCA 构造视图所需要的时间的增长速度也低于 SCM，这是因为 SCVCA 通过主控制节点将构造任务分配给从控制节点来共同构建视图，从控制节点将自己控制的局部网络生成视图后返回主控制节点，这比 SCM 中将所有网络信息都发送给请求节点大大降低了数据传输的时间和构造过程中的计算量，从而有效降低了视图构建的时间。

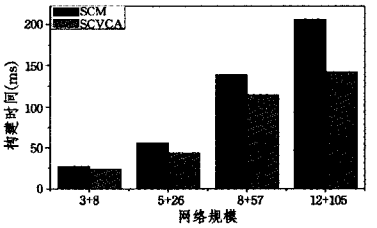

图 6 在不同网络规模下两种视图生成算法的构建时间比较

多个控制节点构建视图必然造成网络传输的负担，进行网络视图构建的算法必须能够具有良好的可扩展性，不能随着网络规模的增长而出现爆炸性的消息传输数量。为了比较 SCVCA 和 SCM 随着网络规模的增大和网络中需要构建视图的节点的数量的增加引起的网络传输负载，给出了两种算法在多种网络规模下和多种需要构建视图的节点比例下引起的网络传输的消息数，如图 7 所示。从图 7 中可以看出，SCVCA 在各种情况下构建视图造成的网络消息数均明显低于 SCM 的，这是因为 SCVCA 由主控制节点负责为全网的所有请求生成一次视图，而 SCM 需要每个控制节点都自己请求网络视图的构建信息来自己构建视图，这势必造成 SCM 构建视图过程中随着网络规模和请求数的增加而造成网络信息传输量的迅速增加。

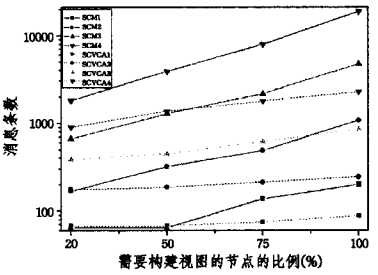

图 7 不同网络规模下两种算法的消息数比较

结束语 本文首先分析了在可信可控网络中利用多个控

（下转第 311 页）

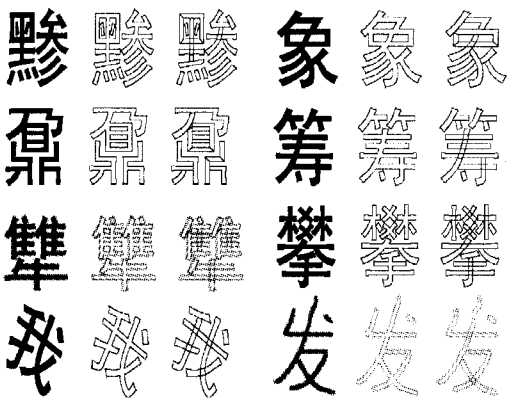

图 8　部分实验结果图

表 4　本算法时间与单字细化时间比较表(时间 ms)

汉字	本算法	细化时间	汉字	本算法	细化时间
黵	328	594	鬓	282	687
象	250	797	群	265	656
鼎	282	735	攀	328	766
筹	281	734	我	218	859
犨	313	812	发	94	78

结束语　不同于基于细化图像和基于原始图像的笔画提取算法，本文提出了基于轮廓的汉字笔画提取算法。该算法首先检测文字的轮廓，然后提取轮廓的特征点，并在特征点中区分出凹点，最后对轮廓进行跟踪，找到笔画相交处的凹点，将两个匹配的凹点连接起来，完成一次轮廓的跟踪后，即可分离出一个完整的笔画轮廓，实现笔画的提取。实验证明该算法对提取轮廓较平滑的字体笔画是有效的。

参 考 文 献

[1] 靳天飞. 脱机手写汉字识别中笔段提取算法研究[J]. 山东大学学报：理学版，2008，43(5)：39-44

[2] Yu Qiao，Mikihiko N，Makoto Y. A framework toward restoration of writing order from single-stroked handwriting image[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，2006，28(11)：1724-1737

[3] L'Homer E. Extraction of strokes in handwritten character[J]. Pattern Recognition，2000，33(7)：1147-1160

[4] 刘峡壁，贾云得. 一种字符图像线段提取及细化算法[J]. 中国图象图形学报，2005，10(1)：48-53

[5] 何浩智，朱宁波，刘伟. 基于骨架点分布规律的汉字笔段提取算法[J]. 计算机工程与应用，2007，43(22)：83-86

[6] 韩丽娟，王勇，耿辉，等. 一种新的边缘检测算法在图像处理中的应用[J]. 仪器仪表与分析监测，2008(4)：18-20

[7] 程立，王江晴，田微，等. 手写体女书文字规范化处理程序研究[J]. 中南民族大学学报：自然科学版，2012，31(1)：93-96

[8] 程立，王江晴，田微，等. 改进 D-P 算法在图像轮廓平滑中的应用[J]. 计算机工程，2012，38(17)：232-234

[9] 袁媛，刘文才. 基于形状分割的手写汉字笔画提取方法[J]. 计算机工程与科学，2010，32(12)：57-60

(上接第 301 页)

制节点对一个 AS 进行联合控制的情况下，为多个控制节点提供一致性视图的必要性，并给出了各个控制节点独自构建网络视图造成的问题；然后针对这些问题，提出了可信可控网络中为多个控制节点构建一致性视图的机制，包括主控制节点的选举算法和基于选举的一致性视图的构建算法，保证了视图版本一致性，并对该算法构建的视图的一致性进行了分析；最后在控制信息数据库的基础上，为控制节点提供了 AS 内设备粒度和协议粒度的一致性视图。

本文解决的一致性视图问题是可信可控网络中多个控制节点对 AS 进行联合控制的基础，然而多个控制节点对 AS 进行有效控制还涉及到 AS 外信息的共享问题。为了更好地提高网络控制的效率，建立多个控制节点 AS 间有效的信息共享机制将是非常有意义的工作。

参 考 文 献

[1] Bouabene G，Jelger C，Tschudin C，et al. The Autonomic Network Architecture(ANA)[J]. IEEE Journal on selected areas in communications，2011，28(1)

[2] Iqbal H，Znati T. Distributed control plane for 4D architecture [C]// the Proceeding of Global Telecommunications Conference. Washington DC，US，2010：1901-1905

[3] 吴杰. 分布式系统设计[M]. 北京：机械工业出版社，2011

[4] King G T，Gendreau T B，Ni L M. Reliable election in broadcast networks[J]. Journal of Parallel and Distributed Computing，2009(1)：521-546

[5] SSF-Scalable Simulation Framework[OL]. http://www.ssfnet.org，2010-04-23

[6] Dolev D，Klawe M，Roth M. An O(n log n) unidirectional distributed algorithm for extrema finding in a circle[J]. Journal of Algorithms，1982(3)：245-260

[7] Zhang In，Perrig A，Zhang Hui. Centaur：A Hybrid Approach for Reliable Policy-Based Routing[C]// the 29th International Conference on Distributed Computing Systems(ICDCS). June 2012

[8] Mans B，Santoro N. Optimal elections in faulty loop networks and applocations[J]. IEEE Transactions on Computers，1998，47(3)：286-297

[9] Wang P，Luo J Z，Li W，et al. Control Information Description Model and Processing Mechanism in the Trustworthy and Controllable Network[C]// Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM09). New York，June 2011：398-405

[10] Wang Pan，Li Si-kun，Cai Xun，et al. The Construction A lgorithm of Binary Interval Tree Nodes based on Lattice Patition [J]. Journal of Computer Aided Design & Computer Graphics，2012，23(7)：1115-1130