CVE环境下一种基于QoS的动态接入控制机制研究

庄 艳[1],陈继明[1,2],徐 丹[1],张凯隆[1],潘金贵[1]
(1.南京大学计算机软件新技术国家重实验室,江苏南京 210093;2.江苏大学计算机科学与通讯工程学院,江苏镇江 212013)

摘 要: 针对协作式虚拟环境多用户协作的特点和需求,本文提出了基于QoS的动态接入控制机制。该方法动态更新系统最小链路延时,用户在加入系统协作时,系统可将用户订购的服务质量与系统最小链路延时进行比对,从而允许用户加入或者被挂起。实验结果表明这种动态接入控制方法,能够稳定网络流量、减轻路由器处理负荷,同时能使用户实时获得系统状况,调节本地处理。

关键词: 协作式虚拟环境; QoS; 动态接入控制; 最小链路延时
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2009) 08-1699-08

Research on Dynamic Admission Control Mechanism Based on QoS in CVE

ZHUANG Yan[1], CHEN Ji-ming[1,2], XU Dan[1], ZHANG Kai-long[1], PAN Jin-gui[1]
(1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu* 210093, *China*;
2. *Shcool of Computer Science and Telecommunications Engineering, Jiangsu University, Zhenjiang, Jiangsu* 212013, *China*)

Abstract: Based on the characteristics and needs of collaboration for multi-users in Collaborative Virtual Environment (CVE), this paper proposed a dynamic admission control mechanism based on QoS. It updated system minimum delay dynamically. When users tried to access the system, it compared the QoS of users' subscription and system minimum delay to determine whether they can access or just hung up. The experiment shows that the dynamic admission control mechanism can make network flow stable and reduce router process load, and let users get the system information at real time to adjust their local processes.

Key words: collaborative virtual environment; QoS (Quality of Service); dynamic admission control; minimum delay

1 引言

协作式虚拟环境(Collaborative Virtual Environment, CVE)是支持分布式协作的多用户虚拟环境[1].它允许来自不同地理位置的用户动态地加入、共享这个虚拟环境,并通过相互协作共同完成某项任务[2].然而,随着用户的不断加入,网络环境的扩大,受限的网络资源很大程度上限制了CVE系统的性能,尤其是网络服务的多样化导致了信息量的不断增大,使得这一问题尤为突出.对此,Zabele和Oliveira针对于分布式虚拟环境的特点提出了主动路由技术[3,4],即由订购方向系统发送订购域,发布消息在路由器上根据订购域进行转发的方式[5].由此减少了与兴趣无关的消息,从而使网络流量得到了一定的控制.在协作环境下,用户以组的方式来共同完成任务,使得他们之间的关系更加紧密,同时对数据链路的传输质量也提出了更高的要求.如何提高网络中的数据交换效率,减少冗余信息的传递,为用户提供满足于他们的数据服务,成为CVE迫切需要解决的问题之一.

目前,网络服务质量(Quality of Service, QoS)作为链路传输的标准,广泛地应用到了直接通讯领域.传统的QoS通过直接链路的网络资源管理来提供所需要的QoS保证.而这种方式较适应直连方式,在松散的间接通讯结构下,无法利用现有的方法提供服务.主动兴趣管理系统是一种间接通讯方式,无法强制进行QoS参数的协商从而提供服务.因此,需要提出适合于这种松散结构的QoS链路服务来满足CVE环境下的多用户协作.本文提出将用户订购延时要求与检测得到的系统最小链路延时动态比对,从而允许或挂起用户接入的动态接入控制机制.经实验表明,动态接入控制机制能减少无效报文的转发,缓解主动路由器上的报文处理负荷和减少网络传输中的延时.

收稿日期:2008-04-22;修回日期:2009-02-14
基金项目:国家自然科学基金(No.60533080, No.60473113);计算机新软件技术国家重点实验室开放课题(No.KFKT2009B16)

2 相关研究

为了解决在满足用户协作需求的前提下,减少通讯量以实现可扩展的问题,从 90 年代 CVE 提出至今,已有不少 CVE 系统提出了它们的解决方法.M Kallmann 和 D Thalmann 最早提出 Smart Object[6]方法,将场景中用于协作的物体封装成为一个 Object.该 Object 本身携带所有能够对其进行的操作以及利用它进行协作的方法.用户在环境中的虚拟对象可以通过调用 Object 自身方法来实现协作,这是一种第三方协作方式.它的问题在于,用户的协作必须通过 Object 来实现,协作行为固化于物体本身,灵活性不强.Pieter Jorissen 为了提高动态特性,又提出了将所有虚拟场景中的物体和用户化身都封装成 Object,携带各自的协作信息,从而实现自由的协作[7].利用 Object 封装协作信息的方式,有效地实现了用户之间的协同工作.基于 Object 的方法在应用层面上实现了 CVE 的协作需求,一定程度上减少了协作用户之间的消息传递,然而并没有从理论和实践上保证服务质量,无法保证协作的准确性.同时,它要求预先定义协作种类和协作行为,无法体现较好的动态特性.采用现有的 QoS 理论,能够较好的反应链路传输状态.从这个角度出发,提升 CVE 协作的服务质量和可扩展性不失为一种更好的选择.引入了 QoS 服务的 MASSIVE-2[8]系统,在基于空间认知模型的基础上,增加了基于 QoS 的视频流服务.在原有系统的基础上,加入 QoS Manager 来将视频流进行分层,各个层次的数据流按照不同的帧速率进行转发,同时采用分组的概念来共享数据流资源.但它的视频流服务层次是在系统设计初就决定的,不能随着网络状况的变化而做出相应的更改.QoS 服务表现在应用层用户请求到底层数据流服务层次的映射上,而没有直接反映到链路层,实时体现网络状况.

同样文献[9]总结了将 QoS 应用于网络虚拟环境的相关问题,提出使用三层映射的方法来实现应用层服务需求到底层链路的数据传输.但依然没有从数据流量上来说明 QoS 服务的实施和所带来的优势.

与此同时,在发布订购领域的发展中,由于协作的加入和系统功能的不断扩展也开始研究使用 QoS 作为质量保证的重要协议.文献[10]中首次给出了发布订购领域中 QoS 参数的全面定义和评估.由文献[12]详细描述了在无线传感器网络发布订购系统中成功引入 QoS 的方法.F Araujo 等人提出将 QoS 参数作为属性的形式,加入到发布订购系统[13,14].最新的文献[11]介绍了一种新的平衡发布订购系统负载的方法.文献[15]中研究了分布式环境下时空一致性的重要性,体现了在协作式环境下提供有效质量保证的需要.

由于 CVE 环境下采用了兴趣区域管理这种松散的网络结构,就目前的研究情况来看,没有直接在链路层提供 QoS 服务,来实现 CVE 数据缩减以及用户服务的保证.基于此并结合了以上领域的最新发展,本文提出了将延时参数应用于 CVE 环境中,使用动态接入控制机制实现控制网络流量和路由器处理负荷的目的,同时使用选举算法来实时更新各个路由器上的链路延时.根据当前的网络链路状况来动态地控制协作用户的加入,从而减少无效发布消息在系统中传输,最终达到减少系统负载的目的.

3 有关定义

将系统中的 n 个主动路由器 AR 记做 $AR_0, AR_1, \cdots, AR_{n-1}$.其中 AR_0 是核心主动路由器.每个主动路由器拥有一个上游端口和数个下游端口用以与其他主动路由器或主机连接.将系统中的 m 个主机记作 $H_0, H_1, \cdots, H_{m-1}$.在进行接入控制时,下游 AR 需要请求方向的上游 AR 发送订购请求延时以及当前请求报文的延时,用以进行链路检测,我们将此消息定义为 $< X, d_{now}, d_{req} >$,d_{now}为请求报文当前的累计延时,d_{req}为订购所要求的端到端的延时请求,其中 $x \in \{AR_i\} \cup \{H_k\}, 0 \leqslant i < n, 1 \leqslant k < m$.端到端的链路延时表示为 $d(x,y)$,即节点 x 到节点 y 的链路延时,$x, y \in \{AR_i\} \cup \{H_k\}, 0 \leqslant i < n, 1 \leqslant k < m$.节点之间的路径长度表示为 $len(x,y)$,$x, y \in \{AR_i\} \cup \{H_k\}, 0 \leqslant i < n, 1 \leqslant k < m$,使用它们之间的链路数来表示长度.假设 a、b、c、d 四个主动路由器依次连接,a 为 b 之上游 AR,b 为 c 之上游 AR,c 为 d 之上游 AR.则 $len(a,d)=3$.

定义 1 主动路由器通过下游端口,向下可达的主机称为该该主动路由器的下游端接点,记为 $dH(AR_i), 0 \leqslant i < n$.其中,系统所有主机均为核心路由器的下游端接点.

定义 2 主动路由器 AR_i 下游所有端节点到达该主动路由器的链延时的最小值称为主动路由器 AR_i 的下游最小链路延时,记为 $d_{min}(*, AR_i)$.表示为:

$$d_{min}(*, AR_i) = \min_{X \in dH(AR_i)} d(X, AR_i) \tag{1}$$

定理 1 设主动路由器 AR_i 从下游虚拟端口接收到接入请求消息 $< x, d_{now}, d_{req} >$,下游存在满足延时要求的链路当且仅当 $d_{now} + d_{min}(*, AR_i) \leqslant d_{req}$,其中,$x \in \{AR_i\} \cup \{H_k\}, 0 \leqslant i < n, 1 \leqslant k < m$.

证明: 设主机 X 向上游发送接入请求,经过 $ar_1 \cdots ar_j$ 到达 AR_i,令 $ar_0 = X$,其中 $ar_j = = AR_i, 1 \leqslant j < n$,$n \in N$.可知,

$$d_{now} = \sum_{k=0}^{j-1} d(ar_k, ar_{k+1}), x = ar_{j-1} \tag{A}$$

充分性:若满足 $d_{now}+d_{min}(*,AR_i)\leqslant d_{req}$,则由定义 2 得,$\exists X'\in dH(AR_i)$使得 $d_{min}(*,AR_i)=d(X',AR_i)$,将此链路记为 $ar_0',\cdots,ar_k'$,其中 $ar_0'=X'$,$ar_k'=AR_i$,$1\leqslant k<n$,$n\in N$.

故,得到的链路 $ar_0,\cdots,ar_{j-1},AR_i,ar_{k-1}',\cdots,ar_0'$,即为满足延时要求之链路.

必要性:存在链路 $ar_0,\cdots ar_{j-1},AR_i,ar_{k-1}'$,$ar_0'$满足延时要求,有 $ar_0=X$,$ar_0'=X'$,$X'\in dH(AR_i)$.则有

$$d(X,X')=d(X,AR_i)+d(AR_i,X')\leqslant d_{req}$$
$$=\sum_{l=0}^{j-1}d(ar_l,ar_{l+1})+\sum_{l=0}^{k-1}d(ar'_l,ar'_{l+1})\leqslant d_{req}$$

由等式(A)进一步化为 $d_{now}+\sum_{l=0}^{k-1}d(ar'_l,ar'_{l+1})\leqslant d_{req}$又由定义 2 可知 $d_{min}(*,AR_i)\leqslant\sum_{l=0}^{k-1}d(ar'_l,ar'_{l+1})$,所以得到 $d_{now}+d_{min}(*,AR_i)\leqslant d_{req}$.综上所述,定理 1 得证. #

定理 2　设主动路由器 AR_i 从下游虚拟端口接收到接入请求消息 $<x,d_{now},d_{req}>$,上游存在满足延时要求的链路必有 $d_{now}+d(AR_i,up(AR_i))\leqslant d_{req}$,其中 $x\in\{AR_i\}\cup\{H_k\}$,$0\leqslant i<n$,$1\leqslant k<m$,$up(AR_i)$为 AR_i 的上游主动路由器.

证明:(反证法)　设主机 X 向上游送接入请求,假设存在链路 $ar_0,\cdots,ar_{j-1},AR_i,ar'_{k-1},\cdots,ar'_0$,满足延时要求,有 $ar'_0=X$,$ar'_0=X'$,$X'\in dH(AR_i)$.

反设在主动路由器上 $d_{now}+d(AR_i,up(AR_i))>d_{req}$,则 $ar_0,\cdots,ar_{j-1},AR_i$ 段链路即已经超过请求延时要求.故该链路的总体延时必定大于 d_{req}.从而假设不成立,得证. #

定理 3　设核心路由器 AR_0 之下游最长支路长度为 l.若主机 H_j 向系统发出接入请求 $<H_j,d_{now},d_{req}>$,则该消息验证的路径不大于 l.

证明:设 H_j 的请求的向上验证链路为 $ar_0,\cdots,ar_{k-1}$,有 $ar_0=H_j$,ar_{k-1}为最后回复 H_j 接入请求之节点.定理 3 则是要证明 $k\leqslant l$.利用反证法证明.反设 $k>l$.则验证链路可表示为 $ar_0,\cdots,ar_{l-1},\cdots,ar_{k-1},ar_0=H_j$.由于 AR_0 最长支路长度为 l,则有 $\forall H_l\in\{H_i\}$,$len(H_l,AR_0)\leqslant l$.而验证链路为 $ar_0,\cdots,ar_{l-1},\cdots,ar_{k-1},ar_0=H_j$ 则其中链路中必定经过 AR_0,进一步将验证链路表示为 $ar_0,\cdots,ar_i,ar_0=H_j$,$0<i<l-1$.在 AR_0 处,若下游验证不通过,则根据定理 1,表明 $d_{min}(*,AR_0)+d_{now}>d_{req}$.由定义 2 可知,$\min_{X\in dH(AR_0)}d(X,AR_0)+d_{now}>d_{req}$,而对于 AR_0,由定义 1 可知,$\forall H_l\in\{H_i\}$,$H_l\in dH(AR_0)$.从而得到系统中每个端接点都无法满足此请求延时需要,因而拒绝接入.故,接入消息无需再向其他节点发送验证,则验证链路修正为 $ar_0,\cdots,ar_i,AR_0$,$0<i<l-1$,验证长度不大于 l,与验证链路为 $ar_0,\cdots,ar_{l-1},\cdots,ar_{k-1}$长度为 k,$k>l$ 相矛盾.

故,假设不成立.定理 3 正确. #

4　动态接入控制机制

我们认为系统无法提供服务的订购主机即为无效主机,假若没有采取拒绝的方式阻止主机的加入,其他主机仍向其发送发布消息,那么这些消息会在主动路由器之间进行转发,而由于链路延时过大,即使这些消息到达订购主机,但却不是主机所期望的,是无效的消息;同时,由于在 CVE 环境中,强调参与者之间的协作,若系统无法满足订购主机的服务要求,也就是无法使其与其他参与者相互协作,即使它发送的发布消息能够满足其他参与者的要求,但仍然被认为是无效的参与者.根据以上两点,我们采用了动态接入控制机制,在主机加入系统时,判断当前网络状况是否能够满足其协作请求,从而接入或者被挂起.这一机制一方面保证了参与者的活跃性,第二保证了发布消息的有效性,同时减少了系统由于链路延时而带来的冗余消息的传送,从而减轻了网络负荷.以下,我们将对这一控制过程进一步描述.

4.1　动态接入控制机制

图 1 给出了系统的简单 4 层拓扑结构图.用圈表示主动路由器,三角表示主机订购端,其中灰色的表示待加入的主机,实线表示已经连接的主机,虚线表示待建立的连接,线上的数值表示该链路的延时状况.

0 号为核心路由器节点,由于是树结构,所以核心路由器节点可以是树中的任何一个节点,只是需要额外完成链路测试和维护功能.对于链路最小延时和直

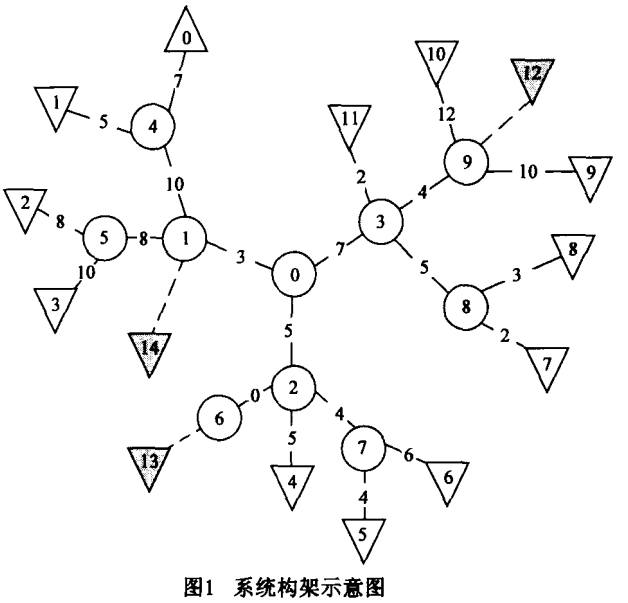

图1　系统构架示意图

连链路延时的动态测试就是由核心路由器负责完成的,这在后面的部分会描述,在此每条链路的延时状态都已经标明.假设主机 H_1 的请求 $d_{req}=8$,主机 H_2 的请求 $d_{req}=12$,主机 H_3 的请求 $d_{req}=14$.它们分别代表了上游验证,启动主动路由器和核心路由器验证三种情况.由于直接接入的情况,在每种情况中都会涉及,故不赘述.以下我们依次讨论它们各自的加入过程.由定理3可知,检测的最大路径是从加入节点到核心路由器,因而我们只需要考虑核心路由器下加入节点分支的链路状况.

首先,在主机 H_1 加入系统之前,系统有 $d(AR_9, H_{10})=12$、$d(AR_9, H_9)=10$, $d_{up}=4$.上游链路延时小于各条下游链路延时的原因是 H_9 与 H_{10}之间发布订购相交多导致了下游链路延时较大,而同时两个端主机的发布消息与 AR_9 裁剪区匹配较少,故上传较少.在这种情况下,路由器 AR_9 主要的数据量主要是下游发布消息的转发,从而导致了这种状况. H_{12}发出 $d_{req}=8$ 请求接入消息,累积延时为 $d_{now}=0$.根据定理1,路由器 AR_9 先查找下游链路是否存在满足条件之链路 $d_{now}+d_{min}(*, AR_9)=10>d_{req}$验证失败;根据定理2,验证上游是否可能存在 $d_{now}+d_{up}=4<d_{req}$,满足,故向上游转发验证 $d_{now}=d_{now}+d_{up}=4$.接着到 AR_3 进行验证,下游验证成功,故返回 accept 消息.若 AR_3 两个验证都未通过,则表明不存在满足条件之链路,返回 deny 消息.消息传递流程见图2.

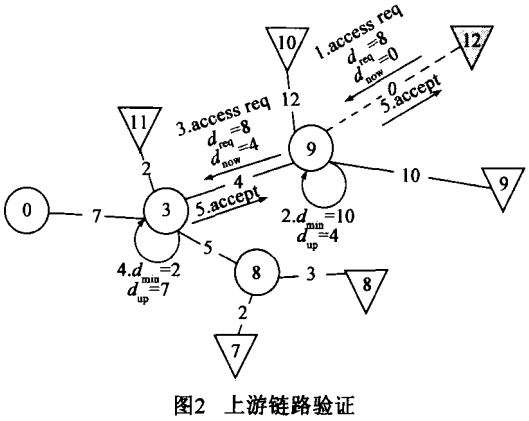

图2 上游链路验证

接着考虑主机 H_{13}的加入情况.主动路由器 AR_6 虽然已经接入系统,但由于其下游并不存在订购端主机,因而认为 $d(AR_2, AR_6)=0$ 即该链路没有数据量,延时为0,且 $d_{min}(*, AR_6)=\infty$.主机 H_{13}向 AR_6请求加入,则 AR_6 继而向上游路由器 AR_2 发送验证申请,其过程与 H_{12}相同,得到 H_{13}允许接入系统.在允许接入之后,修改 AR_6 的最小链路延时为 $d_{min}(*, AR_6)=d_{min}(H_{13}, AR_6)$,此时为0.在之后的系统链路延时更新时,将会更新这个值.

在主机 H_{12}、H_{13}加入之后,我们接着考虑 H_{14}加入的情况.由于路由器 $d_{min}(*, AR_1)=15>d_{req}$且 $d_{now}+d_{up}=3<d_{req}$因此向上游 AR_0 转发接入请求.由于 AR_0 上有系统全局的链路信息,在此验证得到最终的结果.过程如图3所示.

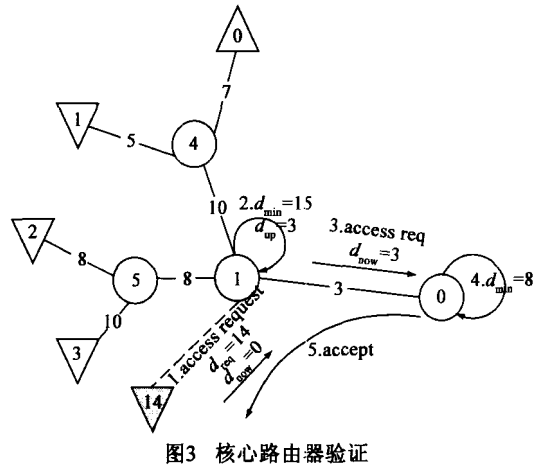

图3 核心路由器验证

综上所述,各个订购主机按照上述过程通过与系统实时链路状况作比较之后,加入或者挂起.通过链路测试算法,每个路由器上都拥有了其下游链路的整体链路状况,从而能够有效地判断下游接入情况.随着核心路由器的实时更新,各个路由器上的下游链路状况信息也将随之更新,从而能够实时根据系统状况较好地判断接入和挂起状况.

4.2 动态接入控制机制的算法过程

主机 H_j 向上游主动路由器 AR_i 发送接入请求 $<H_j, 0, d_{req}>$.(初始化报文延时为0)图4给出了这一算法过程的流程图.

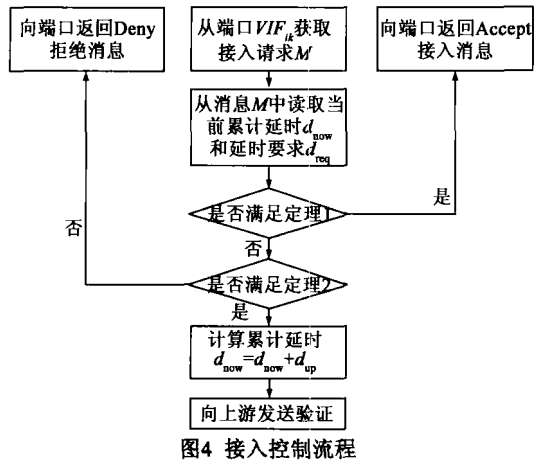

图4 接入控制流程

Step1 AR_i 接收到请求,利用定理1判断下游是否存在有效链路;

Step2 若下游存在满足链路,则直接返回 Accept 消息;

Step3 若下游不存在满足 QoS 要求链路.则根据

定理 2 判断,上游是否存在满足链路;

Step4　若上游存在链路,累积 $d_{now} = d_{now} + d_{up}$,向上发送接入请求 $< AR_i, d_{now}, d_{req} >$,进一步向上验证;若上游不存在,则返回 Deny 消息,拒绝接入.

Step5　当主机接收到 Deny 消息,则调整延时请求或挂起一段时间,再次发送请求.

Step6　当主机接收到 Accept 消息,则表明允许接入,进行第二步订购.向上发送订购消息 $< H_j$, Sub, QoS >.

算法伪代码描述:

主动路由器 AR_i 从下游虚拟端口 VIF_{ik} 接收到接入请求 $< X, d_{now}, d_{req} >$ 之后,第一步接入验证的过程如下:

```
0.    int AdmissionControl( < X, d_now, d_req > )
      {
1.      Get M from interface VIF_ik;
2.      if( d_min( * , AR_i) + d_now < d_req)
        {
3.        send Accept to VIF_ik;
4.        return1;                }
5.        else if ( d_up + d_now < d_req)
          {
6.        M← < AR_i, d_now + d_up, d_req >
7.        Send M to upstream router AR_j
8.        return 0;               }
9.        else Send Deny to VIF_ik
10.         return - 1;
      }
```

若需要向上验证,则主动路由器 AR_i 从转发请求消息之后,将从上游虚拟端口 VIF_{i0} 接收到回复,此回复消息直接向下游请求方转发即可.对于核心路由器上的第一步验证操作,则无需判断上游链路.如下描述:

核心路由器检测函数:

```
0.    boolean AdmissionControlCore()
1.    {     Get M from interface VIF_0k;
2.          if ( d_min( * , AR_0) + d_now < d_req) {
3.            send Accept to VIF_0k;
4.              return true;        }
5.      else send Deny to VIF_0k;
6.                  return    false;          }
```

4.3　链路延时更新机制

动态接入控制的关键参数是系统当前链路状况,主要体现为主动路由器下游最小链路延时和直接相连的链路延时.下游链路最小延时用以动态接入的判断,而直接相连的延时则用以计算累计延时.在我们的设计中,采用核心路由器发送探测报文的方式来获取各个主动路由器最的下游小延时以及下游直接相连链路的延时.对于主动路由器下游最小延时,我们是通过选举算法获得的.简化结构如图 5 所示.

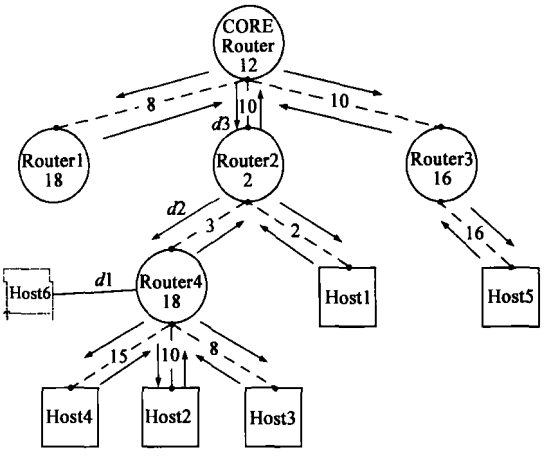

图5　下游链路最小延时

由核心路由器(Core Router, CR)定时向下游链路发送链路探测报文.下游 AR 接收到上游的链路探测报文之后向下游链路转发.直到无下游链路或者达到主机时,向上游回复报文.由于系统是一种分布式的结构,为了避免时间戳问题,我们采取了本地计时的方法.主动路由器从向下分发开始进行计时,直到获得链路回复消息,停止计时获得本条链路的延时情况.对于某个主动路由器 i 其下游最小链路表示为:

$$d_{min}(* , AR_i) = \min_{j \in VIF(AR_i)} d(* , VIF_{ij}) \qquad (2)$$

其中 $VIF(AR_i)$ 表示主动路由器 AR_i 的下游虚拟端口集合,AR_{ij} 表示第 j 个虚拟端口.每个虚拟端口连接一条下游链路,与其他主动路由器相连或者与主机直接相连.通过发送向下发送检测报文开始计时,到收到回复报文获得该下游链路的延时值.由下游所有链路延时值中选取最小值即是本主动路由器的下游最小延时值.与此同时,还需要维护各个主动路由器直连链路的延时值,用于累计延时的计算.同样,我们使用这一探测报文来得出这一值.假设与 VIF_{ij} 的是 AR_k,即 AR_k 是 AR_i 的下游主动路由器,则

$$d(AR_k, AR_i) = d(* , VIF_{ij}) - d_{min}(* , AR_k) \qquad (3)$$

由此方法,在回复的 test 报文中携带 AR_k 的当前下游最小延时,从而上游主动路由器 AR_i 可以计算得到它们之间链路的延时情况.当下游为主机时,它返回的下游最小延时为 0.因此 Test Reply 中含有下游最小链路延时值记为 d_{dm}.链路维护流程如图 6 所示,伪代码表示如下.

```
0. void startTimer()
1. {     if M from upstream is Test;
```

```
2.      {   Timer <—zero;
3.          d_min(*, AR_i) = ∞;
4.          send Test to each downstream VIF;
5.          start Timer;    }
    }
0.  Void computeDelay()
1.    {   Get M_j is Reply_Test from VIF_ij;
2.      d(*, VIF_ij) = Timer.GetTime();
3.      if(d_min(*, AR_i) > d(*, VIF_ij))
4.      d_min(*, AR_i) = d(*, VIF_ij);
5.      d_dn(VIF_ij) = d(*, VIF_ij) - d_dm;
6.          send Test Reply to upstream AR;    }
```

在上述描述中,主动路由器 AR_i 只需要在第一次接收到 Test Reply 消息之后,计算下游的最小链路延时,而后向上游回复.之后到达的回复消息得到的最小链路延时都大于第一个值(此时为集中式时钟).那么,来自于其他下游链路的回复消息则主要用于计算直接相连链路的延时和发现拥塞的情况.简单的例子可以参见图 5 中标注的情况,假设 Host6 进行订购的情况.

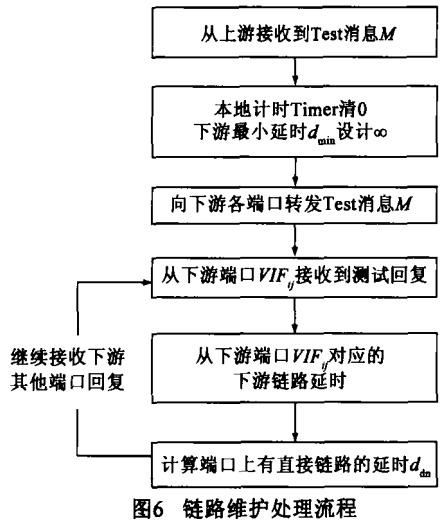

图6 链路维护处理流程

5 性能分析及仿真结果

5.1 性能分析

设需要接入的主机总数为 N,主机间加入间隔为 t,那么在无接入控制情况下 T 时刻系统加入主机数应该为 T/t 记为 N'.设发布消息空间是二维的 $|X|\times|Y|$,即发布消息和订购消息属性是二维的,订购空间表示为 $|X'|\times|Y'|$,用 $\lambda=\frac{|X|\times|Y|}{|X'|\times|Y'|}$ 表示订购空间与发布空间的比例 $\lambda\geqslant 1$,当订购足够多时,两个空间趋向于一致.

定义 3 拥有 N 个主机的系统的发布和订购是均匀的是指任何发布消息与各个主机订购的匹配的概率是相同的,记为 $1/N$.则相对于发布空间来说,匹配的概率为 $1/N\cdot\lambda$.

定理 4 在发布和订购均匀分布以及订购端发布速率相同的情况下,无动态接入控制时,路由器的处理负荷随着直接相连订购主机的增多而增多,从而超过路由器处理能力.

证明:以路由器 A 的处理负荷计算,设其下游直接相连的订购主机数为 x,$1\leqslant x\leqslant N'$,其发布速率为每秒钟 n 个发布消息,则有

$$pNum(A)=n\cdot x+\frac{(N'-x)\cdot n\cdot x}{N\cdot\lambda}=-\frac{n}{N\cdot\lambda}\left(x-\frac{N'+N\cdot\lambda}{2}\right)^2+\frac{n(N'+N\cdot\lambda)^2}{4N\cdot\lambda}$$

上式表示路由器 A 需要处理的报文数.可以看出,当 $x=(N'+N\cdot\lambda)/2$ 时获得最大值.$\because x\leqslant N'\leqslant N\wedge\lambda\geqslant 1$ $\therefore 2x\leqslant N'+N\cdot\lambda$,故 x 在定义域$[1,N']$中单调递增↑.设每个报文的处理代价为 $\Delta cost$,那么 A 的处理负荷为

$$workload(A)=pNum(A)\cdot\Delta cost \tag{4}$$

由上面的计算可以得到最大值为 $x=N'$ 时,$workload(A)=n\cdot N'\cdot\Delta cost$,随着 N 的增加,N' 亦会增加,从而此函数为 N' 的单调增函数↑.当 $workload(A)\geqslant source(A)$ 时,超过路由器 A 的处理负荷,出现丢包、拥塞现象.定理 4 成立. #

定理 5 在发布和订购均匀分布以及订购端发布速率相同的情况下,在无动态接入控制时,路由器的处理负荷随着相连主动路由器的增多而增多,从而超过其处理能力.

证明:以路由器 B 的处理负荷计算,设其下游直接相连的路由器数为 y,$2\leqslant y\leqslant N'$,每个路由器下端接有一台主机,其发送速率为每秒钟 n 个发布消息,则有

$$pNum(B)=y\frac{(N'-1)\cdot n}{N\cdot\lambda}+\frac{(N'-y)\cdot n\cdot y}{N\cdot\lambda}$$

可得当 $y=(2N'-1)/2$ 时,达到最大值,而 y,$2\leqslant y\leqslant N'$,$y\in N$ 且 $y=N'$ 与 $y=N'-1$ 函数结果一致,因而 y 在$[2,N'-1]$上单调递增↑.

进一步得到路由器 B 的处理负荷计算为:

$$workload(B)=pNum(B)\cdot\Delta cost \tag{5}$$

计算得到最大值为当 $y=N'$ 或 $y=N'-1$ 时,$workload(B)=\frac{N'(N'-1)\cdot n}{N\cdot\lambda}\Delta cost$.此函数为 N' 的单调增函数.当 $N'=N$ 即 N' 达到最大后,得到函数 $workload(B)=\frac{N(N-1)\cdot n}{N\cdot\lambda}\Delta cost$.

我们有 $\frac{(N+1)N\cdot n}{(N+1)\cdot\lambda}\cdot\Delta t-\frac{N(N-1)\cdot n}{N\cdot\lambda}\Delta t>0$,故函数对于 N 是递增的.随着 N 的增大,处理器负荷增大,当 $workload(B)>source(B)$ 时,超过本路由器处理能力.

定理 6 在发布和订购均匀分布以及订购端发布速率相同的情况下,采用动态接入控制技术,能够稳定

路由器上的处理负荷.

证明:为简单其见,设订购延时限制为 d.根据链路更新策略,某条链路设为 $H_a, R^c_{i0}, \cdots, R^c_{im}, H_b$, H_a 与 H_b 之间的链路延时应为

$$delay(H_d, H_b) \geqslant \sum_{j=0}^{m-1} trans(R_{ij}, R_{ij+1}) + \max_{j=0}^{m}(workload(R_{ij}))$$

其中 $trans(R_{ij}, R_{ij+1})$表示链路传输时间.

若主机 H_a 带有延时 d 请求能够接入系统,则表明:$\exists H_b, H_a, R^c_{i0}, \cdots, R^c_{im}, H_b, delay(H_a, H_b) \leqslant d$.

由此,该请求实时地检测了链路 $H_a, R^c_{i0}, \cdots, R^c_{im}, H_b$ 各个路由器的 $workload(R_{ij})$状态,从而能够有效地探测到路由器是否超负荷.当请求主机 H 请求接入的主动路由器较均匀时,就能够获得更多更全面的链路信息,从而起到稳定路由器处理负荷的作用. #

定理4、5表明在订购和发布均匀的情况下,随着订购主机的不断加入,路由器会出现超负载的情况,从而导致拥塞的发生.那么,在动态非均匀环境下,局部拥塞的情况会更加严重.同时,定理1、2表明含有QoS要求的订购,在网络延时较大的情况下即使到达订购端也是无效的.定理5则说明了使用动态接入控制技术后,能够实时反映链路上的路由器处理负荷,起到稳定路由器处理负荷的作用,同时也减少了系统链路延时.

5.2 仿真结果

在仿真环节中,我们主要实验证实了接入控制在减少系统负载方面的作用.从系统最小链路延时变化情况以及主动路由器 *AR* 处理负荷的变化情况这两个指标来判定系统负载,分别考察了接入前后主动路由器上这两个指标的变化情况.我们采取了基本的系统构架,由一台核心 *AR*,两台下游 *AR*,以及客户端模拟的两台主机构成,每台客户机模拟50个用户(相互独立的发送订购和发布消息).每个客户端接入之后以20packet/s的速度发送发布报文,每个报文长度为100bytes.在拒绝接入之后采取挂起重试的方法,由于系统链路更新是每3s钟一次,因而将挂起重连时间设定为3s.我们采用随机方法产生订购和发布消息,由客户端进行发送,每个客户采用不同的随机信息.在考察系统负载时,我们读取核心路由器上记录的系统最小延时,并选择其中一台主动路由器上的报文接收速率作为比较指标.如图7所示,在未采用接入控制机制时,随着客户端的加入,*AR* 所接收到的报文数量是不断增加的;在使用接入控制之后,系统初始报文数量增长情况与前者基本一致,而到达600packet/s之后就基本保持在一个稳定的报文流量.

在核心路由器上的链路最小延时情况如图8所示,可以看出在报文数量基本一致的情况下,不使用接入控制的延时仍旧大于接入控制部分的.这是由于在未采用接入控制机制时,由于没有接入控制的发布消息需要往上游转发,从而导致一个发送消息需要向上或向下两个方向进行转发,而引入接入控制之后,由于客户端的订购带有延时要求,因而匹配不满足的报文就可以直接丢弃.同时,接入控制可以较好的将系统链路状况保持在一个相对较低的水平.刻度52~85之间时,未接入控制的系统报文达到800packets/s,从而延时出现了较高的情况;而在接入控制系统下,由于保证了处理报文在600左右,因而有效地控制了系统延时.

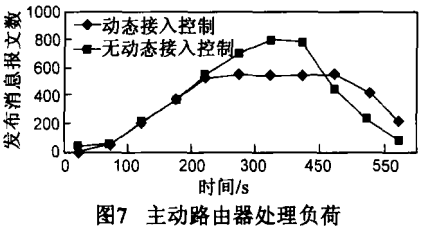

图7 主动路由器处理负荷

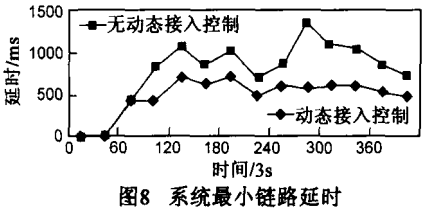

图8 系统最小链路延时

针对CVE领域动态、灵活的特点,本文采用了延时作为参数的接入控制方法来稳定系统处理负荷.而根据不同的领域问题特点,接入控制还能采取速率作为控制条件[16],以及结合了网络安全验证的控制方法[17].前者的覆盖计算虽然能够提供发布订购这种松散结构的服务保证,但由于计算较多不适合于对象变化较多的CVE动态环境.另外在CVE环境中,协作者之间的共享性在安全方面的考虑并不是其主要问题,多步协议影响了系统环境的可扩展性和灵活性.本文这种控制信息更新灵活,接入计算量较小的动态控制方法能够适应CVE环境中灵活、松散的通信结构,同时从动态接入控制方法使用前后的实验对比可以看出其能够实现稳定系统处理负荷的目的.

6 结论

本文针对协作式虚拟环境下协作种类众多,基于发布订购通讯协议的系统结构松散的情况,提出了动态接入控制机制用于提供服务质量保证.这一机制减少和稳定了系统处理负荷,保证了协作过程中消息的有效性和准确性以及系统本身的可扩展性.首先使用动态链路延时更新机制实时更新各个路由器上延时记录,进一步使用动态接入控制机制,将延时参数作为接入控制条件应用于系统的发布订购之中,有效地实现了稳定系统负载和网络负荷的目的.实验结果表明该

方法是可行和有效的.

对于较高的协作请求,需要提供实际的网络资源保证.今后我们将在接入控制的基础上,进一步考察较高协作要求的资源预留问题.

参考文献:

[1] Benford, S D, Bowers, J, Fahlén, L E, Greenhalgh, C M, Mariani, J, Rodden, T R. Networked virtual reality and cooperative work [J]. Presence: Teleoperators and Virtual Environments, Fall 1995, 4(4): 264 - 386.

[2] M Matijasevic et al. A framework for multi-user distributed virtual environments [J]. IEEE Transaction on Systems, Man, and Cybernetics. Part B, 2002, 32(4): 416 - 429.

[3] Zabele S, Dorsch M, Ge Z, et al. SANDS: specialized active networking for distributed simulation [A]. Proceedings of the 2002 DARPA Active Networks Conference and Exposition [C]. Washington, DC: IEEE Computer Society, 2002. 356 - 365.

[4] Oliveria M, Crowcroft J, Diot C. Router level filtering for receiver interest delivery [A]. Proceedings of Networked Group Communication (NGC) 2000 on Networked group communication [C]. New York: ACM Press, 2000. 141 - 150.

[5] 马建刚,黄涛,汪锦岭,徐罡,叶丹.面向大规模分布式计算发布订阅系统核心技术[J].软件学报,2006,17(1):134 - 137.
Ma Jian-Gang, Huang Tao, Wang Jin-Ling, Xu Gang, Ye Dan. Underlying techniques for large-scale distributed computing oriented publish/subscribe system [J]. Journal of Software, 2006, 17(1): 134 - 137. (in Chinese)

[6] M Kallmann, D Thalmann. Modeling objects for interaction tasks [A]. Proceedings of Eurographics Workshop on Animation and Simulation [C]. Lisbon, Portugal, 1998, 73 - 86.

[7] P Jorissen, M Wijnants, M Lamotte. Dynamic interactions in physically realistic collaborative virtual environments [J]. IEEE transactions on visualization and computer graphics, 2005, 11 (6): 649 - 660.

[8] Reynard G, Benford S, Greenhalgh C. Awareness driven video quality of service in collaborative virtual environments [A]. Proceedings of ACM Conference on Human Factors in Computing Systems [C]. Los Angeles: ACM Press, March 1998, 464 - 471.

[9] D Gracanin, Yunxian Zhou, L A DaSilva. Quality of service for networked virtual environments [J]. IEEE Communications Magazine, 2004, 42(4): 42 - 48.

[10] S Behnel, L Fiege, G Muhl. On quality-of-service and publish-subscribe [A]. Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops [C]. Lisboa: IEEE Computer Society, 2006, 20 - 25.

[11] Xiangfeng Guo, Hua Zhong, Jun Wei, Dongli Han. A new approach for overload management in content-based publish/subscribe [A]. Proceedings of the International Conference on Software Engineering Advances (ICSEA) [C]. 2007, 32 - 32.

[12] E Nett, J Kaiser, W Schroder-Preikschat. Providing QoS for publish/subscribe communication in dynamic ad-hoc networks [D]. Department of Distributed Systems (IVS) Otto-von-Guericke-University Magdeburg, 2006.

[13] F Araujo, L Rodrigues. On QoS-aware publish-subscribe [A]. Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCS) [C]. IEEE Press, 2002, 511 - 515.

[14] N Carvalho, F Aráujo, L Rodrigues. Scalable QoS-based event routing in publish-subscribe systems [A]. Proceedings of the 4th IEEE International Conference on Network Computing and Applications (NCA'05) [C]. Washington, DC, USA: IEEE Computer Society, July 2005, 101 - 108.

[15] 彭宇行,张拥军,李思昆.分布式虚拟环境的时空一致性研究[J].电子学报,2005,33(2):313 - 316.
Peng Yu-xing, Zhang Yong-jun, Li Si-kun. Time-space consistency in distributed virtual environments [J]. Acta Electronica Sinica, 2005, 33(2): 313 - 316. (in Chinese)

[16] 郭祥丰,钟华,张文博,李京.一种基于速率的发布/订阅系统的准入控制机制[J].软件学报,2008,19(9):2191 - 2202.
Guo XF, Zhong H, Zhang WB, Li J. A rated-based admission control scheme for content-based publish/subscribe systems [J]. Journal of Software, 2008, 19(9): 2191 - 2202. (in Chinese)

[17] 刘伟,杨林,戴浩,侯滨.一种新的网络接入控制方法及其认证会话性能分析[J].计算机学报,2007,30(10):1806 - 1812.
LIU Wei, YANG Lin, DAI Hao, HOU Bin. A new network access control method and performance analysis of authentication session [J]. Chinese Journal of Computers, 2007, 30 (10): 1806 - 1812. (in Chinese)

作者简介:

庄 艳 女,1983 年生于江苏常州.2006 年毕业于南京大学计算机科学与技术系.现为南京大学计算机系博士生,主要研究领域为分布式虚拟环境和 Agent.
E-mail: zhuangyan@mes.nju.edu.cn

陈继明 男,1977 年生于江苏镇江,现为江苏大学计算机学院博士生,主要领域为分布式虚拟环境、Agent 技术. E-mail: jmchen@ujs.edu.cn