新型的环状层次应用层多播模型

许建真[1,2]　王常华[2]　严正岭[2]
(南京大学计算机科学与技术系　计算机软件新技术国家重点实验室　南京 210093)[1]
(南京邮电大学计算机学院　南京 210003)[2]

摘　要　提出了一种新型的环状层次应用层多播模型 HVRB(Hierarchy Virtual Ring-based),它适合于拥有大规模多播成员的多播组。该模型以环作为拓扑结构,同时引入了虚环[1]的概念。采用了基于动态优先级的层次划分方法使得该环状拓扑结构随时间的变化性能逐步得到提高,并且采取一定的措施使得当出现数据传输错误时可以进行快速的恢复。由仿真试验结果可知该模型具有较高的数据传输率、较高的带宽利用率和较低的控制开销。应用该多播模型可以较好地提高多播的性能。

关键词　应用层多播,虚环,层次,性能

Hierarchy Virtual Ring-based Application-layer Multicast Model

XU Jian-zhen[1,2]　WANG Chang-hua[2]　YAN Zheng-ling[2]
(Department of Computer Science & Technology, State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing 210093, China)[1]
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)[2]

Abstract　This paper presents the analysis, design and performance evaluation of HVRB (Hierarchy Virtual Ring-based) protocol, a new scalable application-layer multicast protocol that establishes rings as its topology among group members, which can be used for large group members. Data transmission topology can be easily implemented on top of the ring, and meanwhile virtual ring concept is involved in this paper. Dynamic layer programming method based on priority is used in the paper, and the ring topology is becoming more and more reasonable with the change of time. Fast-speed recovery can be got when data transmission error occurs. Results demonstrate that HVRB has higher data delivery ratio, higher bandwidth usage and lower control overhead in proportion with NICE. Using p2p network simulator and results show that the new model can improve the performance of multicast fairly well.

Keywords　Application layer multicast, Virtual ring, Hierarchy, Performance

1　引言

应用层多播(Application Layer Multicast, ALM)技术以其特有的优点近年来吸引了广泛的研究,并且得到了广泛的应用,提出了很多的应用层多播协议如 NICE[2], Narada[3], Chord[4], CAN[5]。与 IP 多播不同的是,ALM 不需要路由器提供 IP 层多播功能,端系统之间通过单播(unicast)连接,在应用层建立一个虚拟的 overlay 网络。一些接收者收到数据后,通过单播连接再转发给其他接收者。但是 ALM 也带来了一些很严重的问题,例如对网络资源的无节制的掠夺与浪费等等。于是如何提高应用层多播的效率就成了迫在眉睫需要解决的问题。

本文提出的模型是一种分层的环状应用层多播模型。它不同于树优先[6]和网优先[7]结构。它具有传输效率高的特点,同时采用了基于动态优先级的层次划分方法[8],使系统的结构更加合理高效。

2　HVRB 多播模型

首先介绍一下采用环状结构的优点:1)环中节点的度数为 O(1),即在覆盖网上某个节点的邻居节点的数目是个常数,并且节点的度数与多播组中成员个数无关;2)可以通过令牌的方式实现安全、可靠和完全有序的信息传递。

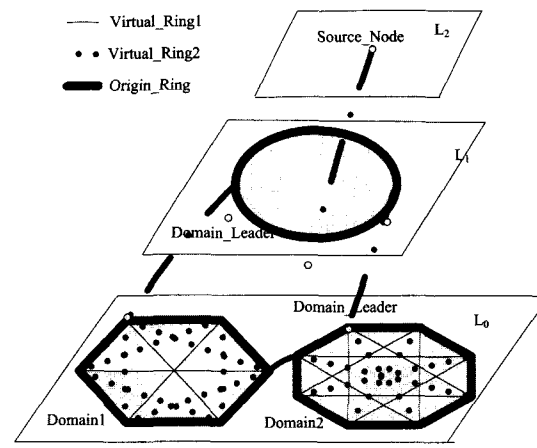

图 1　拓扑结构

该多播模型将多播组成员组织成环状层次结构,共有 $L_0, L_1, \cdots L_{MAX\_NUM\_LAYERS-1}$ 层,所有的多播成员被分到不同的域中,并且所有 L_i 层的域首节点(Domain_Leader)组成 L_{i+1}

许建真　副教授,研究领域为计算机通信与网间互连技术;**王常华**　硕士研究生,研究领域为计算机通信与网间互连技术;**严正岭**　硕士研究生,研究领域为计算机通信与网间互连技术。

层新的域。因此如果一个节点在 L_i 层是域首节点的话，那么在 L_0，L_1，$L_2 \cdots L_{i-1}$ 中也必定是各个域的域首节点。该拓扑结构的最高层只有一个成员节点。在拓扑结构的划分上，L_0 层和 L_1 及以上各层的不同之处在于：L_0 层根据物理位置划分域，由 Rendezvous Point (RP)节点决定域的范围，不限制主机数目。而 $L_1, \cdots, L_{MAX\_NUM\_LAYERS-1}$ 划分域并且要限制域中主机数目在 $k/2 \sim 2k$（k 是常数）之间，当域内主机数目大于 $2k$ 时则进行域的分裂。在 L_0 层要建 Origin 环和 Virtual 环。其余以上各层的节点之间只建 Origin 环。拓扑结构如图 1 所示。

3 拓扑结构的维持

3.1 拓扑结构初始化

起初系统中只有一个 RP 节点，当有节点加入时，向 RP 节点发出加入请求，RP 计算该节点所属的域，并将目前存在的所有 Domain_Leader 信息返还给加入节点。加入节点将 Domain_Leader 信息中的域标志和 RP 返还的域标志进行比较，相同就向该 Domain_Leader 发出加入信息，直接加入到 L_0 层的多播组中。如果找不到相同的域标志说明新加入的节点是某个域中第一个加入多播组的节点，此时 RP 节点负责将该节点加入到多播组中。节点之间通过各自计算出来的优先级的大小（下面的部分将要提到）进行 Domain_Leader 的选择，然后汇报给 RP 节点。

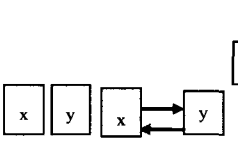

图 2 节点与节点之间合并

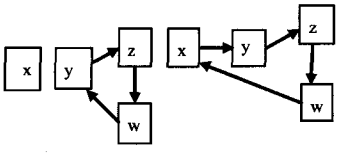

图 3 节点与环之间合并

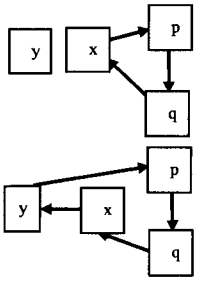

图 4 环与节点之间合并

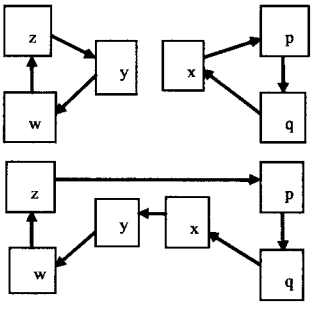

图 5 环与环之间合并

所有的 Domain_Leader 节点每隔 T_{Leader} 时间定时向 RP 节点注册信息，RP 节点返还目前存在的所有其它 Domain_Leader 信息并且刷新该加入节点的信息（如果该节点已经加入到多播组）。然后该 Domain_Leader 向所有其它同一层的 Domain_Leader 发送合并信息，形成更上一层的环状拓扑结构。如果在 $T_{Leader\_lifetime}$ 时间内某个 Domain_Leader 信息没有被刷新，则 RP 认为该域首节点失效，然后删除该节点信息。

L_0 层域内环的形成包括两个方面。一是 Origin 环的建立，在域内主机的加入过程也就是进行环的合并过程。对于环中每个节点都有域首节点及前驱（pred）和后继（succ）节点域，分别存放 Domain_Leader 及 pred 和 succ 节点的 ID 号。当环中只有一个节点时，其前驱和后继均为空，Domain_Leader 域指向自身。合并过程主要包括以下四种情况（x 为申请加入的节点，y 为已经存在的节点）分别如图 2、图 3、图 4、图 5 所示。

当 L_0 层某个域内只有一个 Domain_Leader 时，Origin 环的初始化完成。RP 节点将 Domain_Leader 及其所属的域填入一张映射表中。然后开始进行 Virtual 环的初始化。

另一个是 Virtual 环的建立。Virtual 环的建立比较简单，即在环内的节点之间依据一定的计算公式建立 Virtual Link。

3.2 动态优先级的建立

在拓扑结构的构造过程中，新加入的节点总是在最下层（L_0 层），这对于新节点来说是不公平的。虽然某个节点可能在系统中的存在时间较长，但是由于某种原因使得该节点本身的传输能力下降，结果该节点的下游节点因此不能获得高速的数据传输，这是不合理的。因此采用一种动态分析的方法，根据各节点的资源占有量、时延、带宽、抖动、在系统中的存在时间等，采用加权平均的方法在数据传输过程中，Δt 时间内动态改变拓扑结构中各节点的位置。让拥有大量资源和有效带宽的节点向上游浮动。根据优先级来决定浮动的层数。Δt 的选择主要采用 heart-beat 算法，定期扫描结构中的节点。

设总的资源量为 S，节点在系统中存在的时间为 t，为了精确期间，选用节点的平均时延 ψ 来代替时延。ψ 的值可以通过 M/G/1 排队模型中的 P-K[12] 公式计算得出，如(1)式所示：

$$\psi = 1/(\mu C) + (\rho + \lambda \mu C \sigma_b^2)/[2\mu C(1-\rho)] \quad (1)$$

其中：$1/\mu$ 为报文的平均长度，λ 为报文平均到达率，$\rho = \lambda/(\mu C)$，C 为输出信道的容量，σ_b^2 为服务时间的方差。

在采样点时刻，节点 N 接受的数据为：

$$D_{receive\_data} = \omega D_{origin\_ring\_pred} + \sigma D_{virtul\_ring1\_pred} + (1-\omega-\sigma) D_{virtual\_ring2\_pred} \quad (2)$$

其中：$0 \leqslant \omega, \sigma \leqslant 1$，$\omega$ 和 σ 分别表示数据在 Origin 环和 Virtual_Ring1 环上传输的概率。

节点 N 发送的数据：

$$D_{send\_data} = \tau D_{origin\_ring\_succ} + \xi D_{virtual\_ring1\_succ} + (1-\tau-\xi) D_{virtual\_ring2\_succ} \quad (3)$$

式中：$0 \leqslant \tau, \xi \leqslant 1$，$\tau$ 和 ξ 分别表示数据在 Origin 环和 Virtual_Ring2 环上传输的概率。

则在时间 t 内，节点 N_t 接受的数据（可以在源节点处得到）：

$$D_{receive\_data}(N_t) = S(N_t) \quad (4)$$

发送的数据（忽略传输中速率的变化）：

$$D_{send\_data}(N_t) = \int_0^t D_{send\_data}\, dt \quad (5)$$

故在时刻 t 节点 N_t 的转发效率：

$$\eta = D_{send\_data}(N_t)/\lfloor D_{send\_data}(N_t) + D_{receive\_data}(N_t) \rfloor \quad (6)$$

考虑在某个时刻 t_0，节点 N 的优先级为：

$$P(N_t) = e^{\alpha \times \eta/(1-\eta)} + \beta \times S(N_t)/S + \varepsilon \times t/T + \phi \times 1/\psi \quad (0 \leqslant \alpha, \beta, \varepsilon, \phi \leqslant 1) \quad (7)$$

其中 $\alpha, \beta, \varepsilon, \phi$ 是添加的系数。在不同的网络环境中可以根据实际情况来决定系数的选择。添加系数是为了平衡各参数的权重并让决策者能够操纵节点的运动。在不同的网络状态下选择不同的参数比重可以合理地调整拓扑结构。$S(N_t)$ 为节点 N 的数据资源，T 为系统已经稳定的时间，将优先级划分与系统分层对应，即一定优先级范围的节点应该属于同一层。整个系统稳定后的状况一定是上层节点的优先级比下层高，同层之间则属于同一范围。

3.3 节点加入

节点加入时可以在原来的基础上进行改进，即：为了减轻

RP 节点的负担，假定新加入的主机可以向某个邻近的多播组中的主机 SRP(Sub RP)发出加入请求。由于 SRP 节点掌握了该 SRP 域内所有 Domain_Leader 节点及其域的信息，故使得节点加入变得非常简单。加入过程可以分为两种情况。新节点 New_Host 首先向 SRP 节点发出加入请求，SRP 计算新加入节点所属的域，然后将域信息及所有的 Domain_Leader 信息返还给 New_Host，New_ Host 将自己的域标志和 Domain_Leader 的域标志比较，相同则直接向相应的 Domain_Leader 发出加入消息。没有相同的 New_Host 再向 *RP* 节点发出加入信息。由于 L_1 及以上各层域内主机数在 $k/2\sim 2k$ 之间，由域首节点负责检查域内节点个数，当节点数超过 $2k$ 则要进行域的分裂，分裂时按对半原则进行。当节点数少于 $k/2$ 则相邻的两个域进行合并。域首节点负责域的分裂(合并)，分裂(合并)后另一个域的域首节点由该域中任意一个节点根据计算出来的优先级大小来选择新的域首。

3.4 数据传输

在 Virtual 环的节点 i 与$(i+\lceil\sqrt{N}\rceil)\bmod N$及 i 与$(i+\lceil\sqrt{2N}\rceil)\bmod N$($i$ 为节点的 ID 号，N 为该 Origin 环内节点的总数)之间建立 Virtual Link 用于数据输。发送数据前首先查询节点是否需要数据，如果需要则传输，不需要则不必传输。在 Origin 环上接受到的数据要在 Origin 环和 Virtual 环上同时传输，而在 Virtual 环上接受的数据只在 Virtual 环上传输。节点传送数据的时候，根据 Flooding 算法的原理，对单个节点来说，采用一种有序传输模型，即上层节点先接收数据，接收数据后马上向下层节点传输数据，然后再向后继节点传输。

结论　在 L_0 层使用 Origin 环和 Virtual 环传输数据，网络的直径为 $O(\sqrt{N})$(假设$\sqrt{N}$为整数，N 为环中节点数目)。

证明：假设源节点的 ID 号为 0，分两种情况进行讨论。

情况 1　当域中有一个虚环时，即节点 i 与$(i+\lceil\sqrt{N}\rceil)\bmod N$之间存在虚环。经计算推导，节点 N 的 ID n 可以用下式表示：

$$n=v_0\sqrt{N}+r_0\ (0\leqslant n\leqslant N-1) \tag{8}$$

$$v_0=\lfloor n/\sqrt{N}\rfloor \tag{9}$$

$$r_0=n \bmod \sqrt{N} \tag{10}$$

由上面几式可以得到下式：

$$0\leqslant r_0\leqslant\sqrt{N}-1,0\leqslant v_0\leqslant\sqrt{N}-1$$

其中 v_0 表示在 Virtual_Ring1 环上从源节点 0 到环中某个节点 d 的最短路径所经历的跳数，r_0 表示在 Origin 环上从源节点 0 到环中某个节点 d 的最短路径所经历的跳数。因为 v_0，r_0 的最大值都是$\sqrt{N}-1$，故从源节点 0 到任意一个节点 d 的最短路径最大跳数为 $2(\sqrt{N}-1)$。因此网络的直径为 $O(\sqrt{N})$。

情况 2　当域中有另一个虚环时，即 i 与$(i+\lceil\sqrt{2N}\rceil)\bmod N$之间有环时。经计算推导，节点的 ID n 可以用下式表示：

$$n=v_1\sqrt{2N}+r_1\quad(0\leqslant n\leqslant N-1) \tag{11}$$

$$v_1=\lfloor n/\sqrt{2N}\rfloor \tag{12}$$

$$r_1=n \bmod \sqrt{N} \tag{13}$$

由上面几式可以得到：

$$0\leqslant v_1\leqslant\left(\sqrt{\frac{N}{2}}-\sqrt{\frac{1}{2}}\right),0\leqslant r_1\leqslant\sqrt{N}-1$$

其中 v_1 表示在 Virtual_Ring2 环上从源节点 0 到环中某个节点 d 所经历的跳数。r_1 表示在 Origin 环上从源节点 0 到某个节点 d 所经历的跳数。因为 v_1 的最大值为$\sqrt{\frac{N}{2}}-\sqrt{\frac{1}{2}}$，$r_1$ 的最大值为$\sqrt{N}-1$，故从源节点 0 到任意一个节点 d 的最短路径最大跳数为$\sqrt{\frac{N}{2}}-\sqrt{\frac{1}{2}}+\sqrt{N}-1$，因此网络的直径为 $O(\sqrt{N})$。

综合上述两种情况可以得出使用 Origin 环和 Virtual 环传输数据，网络的直径为 $O(\sqrt{N})$。

当发生传输错误或包丢失的情况下怎么保证数据的有效传输是本模型关心的又一个问题。本模型中提出一种错误恢复方法[9]，即某个节点通过保存若干个相近的，不在同一个域内的环中的节点信息，通过事先计算好的重传序列进行错误恢复，而不依靠上游节点，这样可以避免上游节点发生错误时对下游节点产生的较大影响。具体实现就是一个环中的节点要掌握 p(p 是常数) 个其它环中相邻节点的信息。由于数据是沿着不同的环进行传输，因此不同的环之间同时发生错误的概率会比较小，因此可以在出现错误时能进行快速的错误恢复。

3.5 节点退出

节点退出分成两种情况。一种是节点友好离开，即节点向其前驱和后继节点发送退出信息，附加该节点的后继和前驱信息。如图 6 所示：节点 B 离开，分别向 A 和 C 发送 Good_Bye 消息，并且向 A 发送 C 的信息，向 C 发送 A 的信息。B 离开后 A 的后继变成 C，C 的前驱变成 A。另一种情况是节点不友好地离开。即节点突然失效，那么应由 A(使用 Hello 协议)向 C 发出 Link_Connection 消息，C 收到后回送 A Link_Connection_Ack 消息，A，C 之间建立连接。

如果 Domain_Leader 节点丢失，该域内就没有首节点。为了解决这样的一个问题，规定 Domain_Leader 定期向域内其它主机发出控制信息以证明其存在。如果 Domain_Leader 离开，那么就可以通过该控制信息的丢失识别出来。第一个识别出首节点丢失的主机节点根据节点的优先级来选择新的首节点。

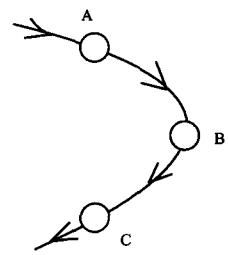

图 6　节点退出

为了防止相邻节点同时失效，可以让一个节点同时保存 m(m 是常数)个后继节点的信息。所有 m 个后续节点同时失效的概率会大大减低，因此可以在很多节点同时离开时进行快速的环的合并。

4 模型的性能分析

4.1 性能参数

1)控制开销(Control Overhead)：形成和维持覆盖网拓扑结构所需的控制信息的数目。利用带宽(bps)的消耗来衡量。

2) 平均数据传输率(Average Data Delivery Ratio, ADDR):节点收到的多播数据包的数目与源节点发送的总的数据包的比值。用式(14)表示:

$$ADDR=\sum_{i=1}^{n} receive(i)/Total \tag{14}$$

$Total$:总的数据传输量;$receive(i)$:节点 i 接收到的数据包数量;n:总的节点数目。

3) 平均链路伸展度(Average Path Stretch, APS):源节点到多播组其它某一节点经过的路径长度与直接通过单播所需经过的路径长度的比值。平均链路伸展度可以用式(15)表示如下:

$$APS=\sum_{r\in R} S_r/|R| \tag{15}$$

其中 S_r 是每个成员节点 r 的伸展率;$|R|$ 是多播组中的成员数目。

4.2 仿真结果

在该部分,将展示仿真的结果以及进行结果的评价分析,最后将 HVRB 同 NICE 进行比较,之所以同 NICE 比较是因为 NICE 协议的控制开销比较大,特别是当节点数目较多的情况下问题会比较严重。

利用 p2psim 在 Linux 平台下进行仿真,节点的加入和退出过程依照泊松过程进行。

图 7 是 NICE 与 HVRB 在控制开销方面的比较。Control Overhead 比 NICE 要低。因为本模型中节点只需要掌握前驱及后继节点及 Domain_Leader 的信息,而 NICE 中的节点要掌握本簇内所有其它节点的信息。

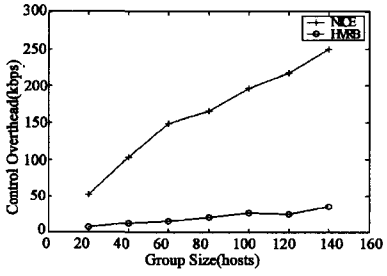

图 7 NICE 和 HVRB 控制开销的比较

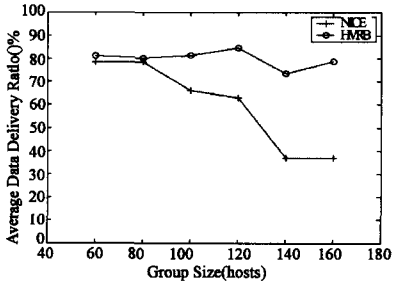

图 8 NICE 和 HVRB 平均数据传输率比较

由图 8 可以看出当有节点失效时 HVRB 的 Average Data Delivery Ratio 比 NICE 要高。这是由于该模型中采用 NICE 分层结构,故当上层节点失效或离开时对下层节点影响比较大,要进行拓扑结构的重构。但本论文中节点可以直接加入到 L_0 层,加入过程非常简单,并且 L_0 层不限制域内节点数目,故省去了最底层域的分裂造成的开销,同时采取的错误恢复算法可以在节点失效时进行快速的错误恢复,减少拓扑结构重构对节点带来的影响,提高了数据传输率。

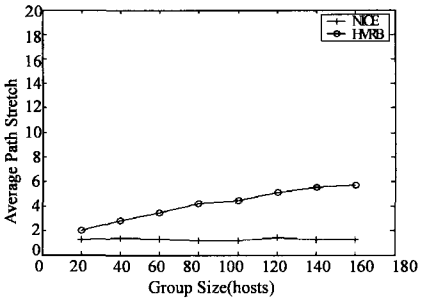

图 9 NICE 和 HVRB 平均链路伸展度的比较

但是 HVRB 在平均路径伸展度(Average Path Stretch)方面比 NICE 还是会高,如图 9 所示。这是因为 HVRB 的节点只有三个后继节点(一个是 Origin 环上的,另外两个是 Virtual 环上的),因此转发数据包最多只能有三次,而 NICE 则不同,它转发数据包的次数由多播树中的相邻节点来决定。

结束语 随着应用层多播在多媒体方面的应用[10]越来越广泛,基于多播的模型不断地被提出。本文提出了一种分层的环状多播模型,实现了数据的分层传输和转发,节省了带宽,采用区域划分法可以实现节点的快速加入和退出。采用基于优先级的动态分析方法,使整个系统的效率趋于合理。引入了 SRP 节点,并且将节点加入和拓扑结构重组时的计算放在了终端节点上,可以减轻 RP 节点的负担。在错误处理方面,采取相应措施使得出现错误后可以快速地恢复。

由于本模型中考虑节点的加入等一些问题都是建立在可信任的角度上的,故下一步的工作重点在于对节点的身份认证及数据在传输过程中的安全问题。由于 HVRB 建立在环状拓扑结构上,故存在路径压力较大和路径伸展度较高的问题,因此 HVRB 还应再采取其它的措施来降低路径压力和路径的伸展度,故降低路径压力和路径的伸展度也是下一步的工作重点。

参 考 文 献

[1] Sobeih A, Yurcik W, Hou J C. VRing: A Case for Building Application-Layer Multicast Rings (rather than trees) // Proceedings of The IEEE Comp-uter Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. 2004:437-446

[2] Banerjee S, Bhattacharjee B, Kommareddy C. Scalable Application Layer Multicat // Proceedings of ACM Sigcomm. Aug. 2002:205-217

[3] Chu Y-H, Rao S G, Zhang H. A Case for End System Multicast //Proceedings of ACM SIGMETRICS. 2002,20(8):1456-1471

[4] Stoica I, Morris R, Karger D, et al. Chord: a scalable peer-to-peer lookup protocol for Internet applications // Proceedings of ACM Sigcomm. 2003,11(1):17- 32

[5] Ratnasamy S, Francis P, Handley M, et al. A scalable content-addressable network. Ph. D. Thesis. University of California, Berkeley, October 2002

[6] Pendarakis D, Shi S, Verma D, et al. ALMI: An application level multi-cast infrastructure // Proc. of the 3rd Usenix Symp. Internet Technologies and Sys. (USITS 2001). SanFrancisco, CA, Mar 2001:49-60

[7] Chawate Y. Scattercast: an architecture for internet broadcast distribution as an infrastructure service [D]. USA: University of California, Berkeley, 2000

[8] Jin X, Wong W-C, Chan S, et al. Serving dynamic groups in application-level multicast // High Performance Switching and Routing, 2005. HPSR. 2005 Workshop. May 2005: 432-436

[9] Yiu W, Wong K, Chan S, et al. Lateral error recovery for media streaming in application-level multicast. Multimedia, IEEE Transactions, April 2006, 8(2): 219-232

[10] Tian R X, Zhang Q, Xiang Z, et al. Robust and efficient path diversity in application-layer multicast for video streaming. Circuits and Systems for Video Technology. IEEE Transactions, 2005, 15(8): 961-972

[11] Xie X R. Computer Network. 4th edition. Beijing: Publishing House of Electronics Industry, 2005: 412-419

(上接第 105 页)

5 对比实验与分析

利用实验室前期开发的 Linux 平台下集群中间件 RTCS,在局域网环境中搭建了模拟实验平台,进行了对比验证实验。

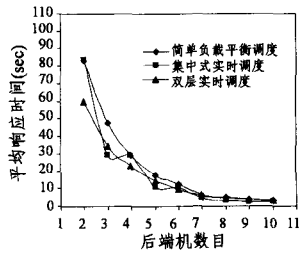

图 5 传输率在 1.54kbps 下三种调度的平均响应时间对比

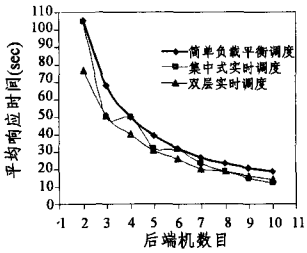

图 6 传输速率在 5.84kpbs 下三种调度的平均响应时间对比

实验的目的主要是考察双层调度机制与一般的简单负载平衡调度在时间响应上的优势。以及与现有集中式实时调度算法在资源利用率方面的比较。

表 1 响应时间下降百分比(与简单负载平衡调度比较)

后端机数	2	3	4	5	6	7	8	9	10
集中调度	0.72%	19.50%	0.25%	16.33%	0.47%	15.23%	17.96%	27.64%	34.24%
双层调度	22.21%	24.16%	21.23%	24.45%	18.65%	25.78%	21.67%	23.55%	26.78%

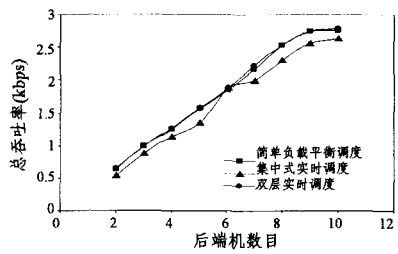

图 7 传输速率为 5.84kpbs 的三种调度策略吞吐率对比

由图 5,6 和表 1 可以看出,两种实时调度算法在时间响应方面明显优于负载平衡算法,其中集中式调度策略在小数目后端机情况下的波动比较大,呈现交替状态。这可能是验证程序在集中式调度模式下存在某一方面的不足导致了性能表现的不稳定。同时也表明,前端机全部承担调度任务,加大了前端机设计与实现的复杂性。然而,双层调度算法则相对来说表现平稳,说明双层调度策略分解了系统整体设计的复杂性。从表 1 中我们还看到,当节点数为 10 的时候,双层调度策略的实时响应性与集中调度相比较略有下降,证明了集中式调度算法的在实时保证方面的优越性。为了验证双层调度策略的整体优势,我们还进行了吞吐率测试。结果如图 7 和表 2。

表 2 总产出率变化百分比(与简单负载平衡算法比较)

后端机数	2	3	4	5	6	7	8	9	10
集中式	−0.34%	−5.57%	−5.42%	−6.22%	0.93%	−6.63%	−8.32%	−8.28%	−4.80%
分布式	0.43%	0.20%	2.02%	1.19%	0.98%	3.23%	0.32%	−0.01%	0.67%

由这组实验我们可以清晰地看出,集中式实时调度的吞吐率明显小于简单负载平衡调度,双层调度机制又显示出明显优势,总体平均比简单负载平衡调度提高了 1.003%的吞吐率。

由此可以得出:双层调度策略不但有效地缩短了系统响应时间,而且在系统资源利用率方面大大优于现有的集中式实时调度策略。这也说明了这种调度策略对于实时集群系统的设计具有重要的参考价值。

参 考 文 献

[1] Buyya R. High Performance Cluster Computing: Architectures and Systems, Volume I. New Jersey: Prentice Hall PTR, 1999

[2] Aron M, Druschel P, Zwaenepoel W. Cluster Reserves: A Mechanism for Resource Management in Cluster-based Network Servers // Proc. of 2000 ACM SIGMETRICS Intl. Conf. on Measurement and Modeling of ComputerSystems. Santa Clara, CA, June 2000

[3] Pradham P, Chiueh T. Implementation and Evaluation of a QoS-capable Cluster-based Router // Proc. of Super Computing 2000. Dallas, Nov. 2000

[4] Bianchini R, Carrera E V. Analytical and experimental evaluation of cluster-based network servers // World Wide Web Journal, 2000, 3(4)

[5] www. isi. edu/research. html # Sensor_Networks

[6] Martinez K, Hart J K, Ong R. Environmental Sensor Networks. IEEE Computer, 2004, 37(8): 50-56

[7] http://os. rdxx. com/Linux/2003-10/4/175020874_2. shtml

[8] Zhang W. Linux Virtual Server for Scalable Network Services // Ottawa Linux Symposium 2000. http://www. linuxvirtualserver. org/Documents. html

[9] Tang R, Simha R. A Delay Differentiation Approach to Real-time Scheduling on Cluster-based Multimedia Servers // ICT 2002. Beijing, June 2002

[10] Nimmagadda S, Harari I. Scalability Issues in Cluster Computing Operating Systems // Proc. of the First Workshop on Cluster-based Computing. Rhodes, Greece, 1999