一种以服务器为通信节点的数据中心网络设计

尹 栋 慕德俊 戴冠中
(西北工业大学自动化学院 西安 710072)

摘 要 根据虚拟网络在数据中心(Data Centers)网络的发展和研究现状,提出了一种基于工业电信网络圆柱型结构的数据中心网络架构。深入阐述了数据中心虚拟网络的结构、网络特性、路由策略以及构建成本等,着重分析了静态路由策略以及基于容错机制的路由算法设计与实现。利用常用 TCP/IP 协议构建网络内部传输模型,测试数据中心网络的传输性能、路由查询速率以及静态路由策略工作效率并进行分析,同时仿真测试了网络的负载平衡及其容错性。

关键词 数据中心网络,虚拟路由器,路由策略,容错性
中图法分类号 TP393.02 **文献标识码** A

Research on Server Centric Data Center Network Design

YIN Dong MU De-jun DAI Guan-zhong
(School of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Data centers are becoming increasingly important infrastructures for many essential applications such as data intensive computing and large-scale network services. According to the previous researches, this paper presented a highly scalable cylinder data center interconnection architecture, with a simple yet efficient routing scheme that maintains short path length in servers communications. It proposed fault tolerant and traffic-aware routing schemes to ensure balanced load. Experiments were used to test the transmission and routing performance of the design. Also, fault tolerance and load balance were evaluated.

Keywords Data center networks, Virtual router, Routing algorithm, Fault tolerance

1 引言

数据中心拥有庞大的机群,已经广泛应用于数据存储、数据分析以及超大型网络服务中[1,2]。在实现千兆兆级数据存储过程中,数据在网络中的传输和处理过程将消耗很多资源,如网络带宽、存储空间以及电能等。在实际应用中,如网页搜索、医学图像处理、社区网络挖掘等领域,数据存储和处理的数量相当庞大,因此对于数据中心来说,一个最基本的要求就是其必须具有很好的扩张性,拥有成千上万甚至几十万数量的服务器。尽管普通计算机成本降低,但是架构庞大数量服务器的机群仅满足扩展性的要求且有效控制整体成本仍是一个难题。由于在数据中心服务器的数量可以急剧增加,因此在架构内部通行的带宽也必须实现大于线性关系或者平方关系的增长,以适应频繁的数据访问和分布式数据处理及存储。为了减少搭建内部链接的成本,通常使用带网卡的普通计算机以及已有网卡的通用服务器。至今为止,构建数据中心内部链接的方式大致可以分为两种:(1)以交换机为中心的结构。通过交换机实现内部连接和服务器数量的扩展,并且不对服务器的配置进行任何修改[3-5]。(2)以服务器为中心的结构。每个服务器仅需要实现数据处理和存储,也需作为数据的转发节点,保证无需对交换机配置进行更改[6-8]。

由于数据中心网络中节点数量巨大,需要建立网络内部统一编址和路由策略,以实现节点之间的高速通信过程,因此人们引入网络虚拟化技术,实现节点资源构建,以及组建和管理规模庞大的网络结构。本文结合网络虚拟化平台的研发经验,将提出一种以服务器为中心的数据中心结构设计方法——圆柱型数据中心网络(Cylinder Data Center Networks, CDCN)。在此架构中,采用以服务器为中心的设计方法,每一个节点由一台服务器组成。与以交换机为中心的数据中心网络相比,CDCN 采用的设计方法更具优势。首先,将网络传输控制由交换机转移到服务器中执行,提高了网络架构中各虚拟网络结构和路由策略的可控性和编程性。其次,以服务器为中心的设计方法采用低端的、网络第二层路由处理的 layer-2 交换机,因此网络架构的成本比较低。

2 CDCN 架构设计

2.1 网络结构

CDCN 网络由两种设备组成:具有两个网络接口的服务器和拥有 n 个端口的交换机,且服务器和交换机被分为 k 列。在该结构中,只有两个参数 n 和 k,因此 CDCN 网络也可称为 (n,k) CDCN。我们用 $H_0 \cdots H_{k-1}$ 表示 k 个服务器列。k 个服务器列和 k 个交换机列在水平方向上环绕成一个圈,如图 1

到稿日期:2011-04-17 返修日期:2011-07-23 本文受国家自然科学基金(60803158)资助。
尹 栋(1982—),男,博士生,主要研究方向为计算机网络控制、信息安全,E-mail:yindong@mail.nwpu.edu.cn;慕德俊(1963—),男,教授,博士生导师,主要研究方向为信息安全、网络控制;戴冠中(1937—),男,教授,博士生导师,主要研究方向为网络控制。

所示。由此可见，CDCN 的结构是一个圆柱型，在水平方向是一个圆环，在竖直方向上由服务器和交换机层相互间隔交错组成。其中，服务器的两个端口分别连接在两侧的交换机上。如架构图所示，在 H_i 列中的服务器，一端连接在 S_i 列的交换机上，另一端则与 $S_{(i+k-1)\%k}$ 列的交换机相连。对于 S_i 层中的一个 n 端口交换机，半数端口连接 H_i 列中的 $n/2$ 台服务器，另一半则连接 $H_{(i+1)\%k}$ 列中的 $n/2$ 台服务器。为了更简单地描述 CDCN 的结构，下面称服务器 $H_{(i+1)\%k}$ 列为 H_i 列的顺时针方向相邻列，而 $H_{(i+k-1)\%k}$ 列则是 H_i 列的逆时针方向相邻列。

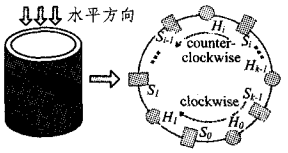

图 1　水平方向上 CDCN 的横截面结构

在拥有 k 列的 CDCN 架构中，每一个服务器列由 $(n/2)^k$ 个服务器构成，每一个交换机列由 $(n/2)^{k-1}$ 个交换机组成。对于 H_i 列的 $(n/2)^k$ 个服务器，用 $(v^{k-1}\cdots v^0)$ 表示，其中 $v^i \in [0,\frac{n}{2}-1](0\leqslant i\leqslant k-1)$。因此，在 CDCN 中一个服务器的位置标识即 ID 号可定义为 $(C,v^{k-1}\cdots v^0)$，其表示在服务器列 H_C 中标号为 $(v^{k-1}\cdots v^0)$ 的服务器。在实现对服务器进行标识的情况下，服务器和交换机之间的内部连接如下：对于在任一服务器列 H_C 中 $2(n/2)^k$ 台服务器及其顺时针相邻的服务器列 $H_{(C+1)\%k}$，可分为 $(n/2)^{k-1}$ 个单元组，且每个组拥有 n 个服务器。在单元组中，每个服务器的标号具有以下属性：当把第 C 列的标识删去之后，其余的标识信息一致。因此，在一个单元组中，一半数量的服务器属于 H_C 列，另一半属于 $H_{(C+1)\%k}$ 列。同时，此 n 台服务器连在同一个标号为 S_C 的交换机上。因此，对于编号 $(v^{k-1}\cdots v^C\cdots v^0)$ 的服务器中，有 $n/2$ 台标号为 $(v^{k-1}\cdots v^C_*\cdots v^0)$ 的服务器属于 H_C 列，其中 $0\leqslant v^C_*\leqslant \frac{n}{2}-1$；另有 $n/2$ 台相同标号的服务器属于 $H_{(C+1)\%k}$ 列，且都通过 S_C 列的交换机连接。

如图 2 所示，在一个(8,2) CDCN 网络结构中，每个服务器列由 $(8/2)^2=16$ 台服务器组成，且每个服务器的编址有两位标号。图中第一行的服务器标号都为(00)，而最后一行的标号均为(33)。若令 $(v^1v^0)=(00)$，那么在 H_1 列中共有 4 台服务器，它们的标号分别是 (v^1_*0)，$0\leqslant v^1_*\leqslant 3$，也就是(00)，(10)，(20)和(30)。同样，在 H_0 列也有 4 台标号为(00)，(10)，(20)和(30)的服务器。这 8 台服务器连在 S_1 列的同一交换机上。

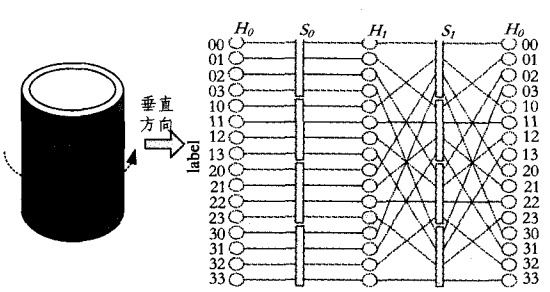

图 2　CDCN 结构的二维平面展开

2.2　网络拓扑结构扩展性

2.2.1　服务器数量

在此结构中，每列有 $(n/2)^k$ 台服务器，在 (n,k)CDCN 网络中总共有 $k(n/2)^k$ 台服务器。定义 N 为 (n,k) CDCN 网络中的服务器总数，那么 $N=(n/2)^k$。若采用 48 端口的 Gbit 带宽的交换机，构建 3 层(48,3) CDCN 架构，那么该架构将有 41472 台服务器。而当扩建到(48,4) CDCN 架构时，服务器的数量将增至 1327104 台；若再在此基础上增加一层结构，即(48,5) CDCN 网络，那么该架构拥有的服务器数量将扩展至 4 千万台左右。由此可见，CDCN 通过简单的链接实现了较好的扩展性。

2.2.2　交换机的数量

在 (n,k) CDCN 网络结构中，每一个交换机列拥有 $(n/2)^{k-1}$ 台交换机，总共需要 $k(n/2)^{k-1}$ 台交换机。在定义的服务器标识 $(v^{k-1}\cdots v^C\cdots v^0)$ 中，若 v^C 保持标号不变，那么总共有 $(n/2)^{k-1}$ 个连接方式，且每一个连接方式需要一台交换机，即一台交换机连接 n 台标号为 $(v^{k-1}\cdots v^C_*\cdots v^0)$ 的服务器(其中 $0\leqslant v^C_*\leqslant \frac{n}{2}-1$)，因此构建 (n,k) CDCN 网络共需要 $k(n/2)^{k-1}$ 台交换机。

3　路由策略设计

3.1　数据包的两步传输

在 CDCN 网络中，包传输过程可以分为两个步骤：第一步是将数据包从源服务器发送到与目的地址一样的中间服务器；第二步是将数据包从中间服务器传输到目的服务器。例如，源服务器 s 发送一个数据包到目的服务器 d，且两服务器的编址分别是 (C_s,L_s) 和 (C_d,L_d)，其中 L_s 和 L_d 为服务器 s、d 的地址标识，$L_s=(v_s^{k-1}\cdots v_s^0)$，$L_d=(v_d^{k-1}\cdots v_d^0)$，其中 $L_s=L_d$。其传输过程如下。

(1)步骤 1：螺旋方向传输

在 H_C 列中的服务器 s 将数据包发送到相邻服务器列 $H_{(C+1)\%k}$ 中的服务器 s_1，这两个服务器的地址标识除第 C_s 位不同，其余均与 s 相同，且 s_1 的标识可以是 $[0,\frac{n}{2}-1]$ 中的任意值。或当 s_1 传输数据包到 s_2，s_2 除了第 $((C_s+1)\%k)$ 位不同，其余均与 s_1 相同，且 s_2 的标识可以是 $[0,n/2-1]$ 中的任意值。此时，在传输过程中，地址标识仅更改一位。因此，当数据包从一个服务器列传输到顺时针相邻的服务器列中时，其地址标识仅变化一位，并且经过 k 次节点跳跃可以达到任何一个服务器。例如，在 (n,k) CDCN 的网络结构中，数据包从 $(0,0\cdots0)$ 传输到 $(k-1,1\cdots1)$ 的途径为 $(0,0\cdots0)\rightarrow(1,0\cdots01)\rightarrow(2,0\cdots11)\rightarrow\cdots\rightarrow(k-1,1\cdots1)$。由于此路径构成了一个螺旋结构，因此称其传输过程为“螺旋方向传输”。

在落选方式传输中，数据包既可以向顺时针方向传输，也可以往逆时针方向传输。但是，在传输过程中，为了避免传输环路，所选择的方向必须前后一致。为此，利用数据包包头中一部分区域来记录传输的方向信息。在实现中，默认的螺旋传输方向为顺时针方向。

(2)步骤 2：环方向传输

当传输到与目的服务器 d 地址标识相同的服务器 d^* 之后，数据包将按顺时针或者逆时针方向传输到相邻的、具有相

同标识 L_d 的服务器列。在实现过程中，CDCN 选择更短的路径进行传输。例如，在 $H_{C_d^*}$ 列服务器 d^* 中传输数据包到 $H_{(C_{d^*}+1)\%k}$ 列的服务器中，此时 $(C_d+k-C_{d^*})\%k \leqslant \lfloor k/2 \rfloor$；否则，服务器 d^* 将数据包发送至 $H_{(C_{d^*}+k-1)\%k}$ 列。但不论往哪个方向传输，下一跳服务器的地址编址必为 L_d。在此过程中，数据包传输路径犹如环状，因此称该阶段为环方向传输。

3.2 静态路由算法

CDCN 采用相对简单的路由算法实现架构内部数据的有效传输，其路由过程如算法 1 所示。源服务器地址为 (C_s, L_s)，目的服务器为 (C_d, L_d)，D 表示传输的方向，$D=1$ 表示顺时针方向传输，$D=-1$ 表示逆时针方向传输，在默认情况下 $D=1$。算法的输出是下一跳服务器地址 (C_u, L_u)。

算法 1 Static Routing Algorithm: SRoute(C_s, L_s, C_d, L_d, D)

Input: (C_s, L_s)表示当前运行路由算法的服务器地址；(C_d, L_d)表示目的地址；D 表示数据包头中记录的传输方向信息(顺时针为 1 或者逆时针为 −1)；其中，$L_s=(v_s^{k-1}\cdots v_s^0)$，$L_d=(v_d^{k-1}\cdots v_d^0)$。

Output: 下一跳服务器地址 (C_u, L_u)。

```
1 if {(Cd+k−Cs)%k≤1 and Ls−v_s^Cs = Ld−v_d^Cs} or{(Cs+k−
  Cd)%k≤1 and Ld−v_d^Cd = Ls−v_s^Cd} then
    /* Ls−v_s^Cs 表示删去标识 Ls 第 Cs 位 */
2       Cu←Cd; Lu←Ld
3   else /* S不能直接达到目的地址 */
4       if Ls==Ld then Lu←Ld; /* 平面环型方向上 */
5           if (Cd+k−Cd*)%k≤⌊k/2⌋) then
6               Cu←(Cs+1)%k
7           else Cu←(Cs+k−1)%k
8           else Ls←(v_s^{k-1}⋯v_d⋯v_s^0); /* 竖直螺旋方向上
*/
10              Cu←(Cs+D−1)%k ;
11 return (Cu, Lu);
```

在路由算法 1 中，并没有计算在两服务器之间的最短路径。但是，算法 1 产生的路径长度是有上限的。在此，对算法中可能产生的最长路径进行分析。在此过程中，将连接在一个 Layer-2 交换机中的两个服务器之间的包传输视为一次跳跃。从源服务器 s 出发，数据包最多经过 k 次跳跃，先通过螺旋方向传输，到达服务器 d^*（服务器 d^* 与目的服务器 d 拥有相同的地址标识）。然后，经服务器 d^* 出发，在环型方向经过最多 $\lfloor k/2 \rfloor$ 次跳跃，达到目的服务器 d。由此可见，在 (n,k) CDCN 架构中，路由算法 1 产生的最长路径为 $k+\lfloor k/2 \rfloor$。

3.3 容错路由策略

3.3.1 在竖直螺旋方向中的容错方法

在螺旋方向的途径中，CDCN 通过迂回的传输方式绕过出错节点，将数据包发送到与目的地址标识相同的服务器。之后，在环型方向上传输数据包到达目的服务器。在螺旋方向上的容错过程可分为 3 个步骤：如果在顺时针方向下一跳服务器出错，那么首先将数据包传输到顺时针方向相邻服务器列；如果在顺时针方向下一跳服务器列中服务器均出错，那么数据包将掉头，向逆时针方向传输；在调转传输方向之后，数据包将一直沿着逆时针方向以螺旋式路径传输。

3.3.2 在水平环型方向中的容错方法

环型传输过程仅有两个方向：顺时针和逆时针。当数据包沿着一个方向传输过程遇到出错节点时，将改变传输方向，沿着相反的方向传输。为了避免产生传输环路，传输方向仅能改变一次。若数据包在沿着改变之后的方向传输中还遇到出错节点，CDCN 节点将丢弃该包。

4 网络性能测试

4.1 单节点性能

在测试过程中，采用普通计算机作为 CDCN 中网络服务器，实现路由策略以及数据转发。计算机配置为 Intel 2.8 GHz 双核 CPU 及 2GB 内存。服务器 P 是 CDCN 架构中的网络节点，实现服务器 A 与 B 之间的通信。

在服务器 P 中，为每一个数据包计算路由信息所需要的 CPU 时钟周期，并进行统计分析。通过测试，CDCN 中静态路由算法平均需要 90 个 CPU 周期计算出一个数据包的下一跳地址；而容错路由算法则需要大约 250 个 CPU 周期为一个数据包获得下一跳地址。为与一般的 IP 查询过程进行对比，在相同测试平台中，服务器 P 运行 Click Router 程序进行 IP 路由查询，实验结果如图 3 所示。当路由表中只有 128 条路由信息时，IP 路由器需要 600 多个 CPU 时钟周期来获得一个数据包的下一跳地址信息；当路由表信息量增大时，IP 路由器将需要更多的时间完成路由查询工作。

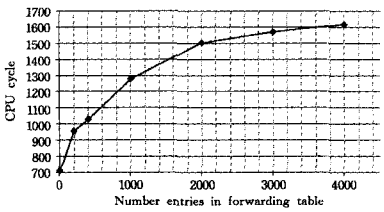

图 3 进行一次 IP 路由查询所需 CPU 时间开销

4.2 平均路径长度

在此实验中，我们采用 12 个端口的交换机，即 $n=12$，且服务器的列数是可扩展的。在测试网络性能过程中，选取 100,000 个随机源与目的地址对，仿真数据包在实际网络中的路由和传输情况。图 4、图 5 描述了平均路径长度的实验结果。在服务器不断扩展过程中，路由过程的平均路径长度与服务器层数呈线性增长关系。对于层数为 k 的 CDCN 网络架构，仿真实验中的平均路径长度与最短路径相差不大，尤其是当服务器列数较少的时候；同时，与最长路径 $k+\lfloor k/2 \rfloor$ 相比，一般平均路径缩短了 20%，实验结果如图 4 所示。随着出错的服务器数量不断增长，路由中平均路径的长度也随之增加，但是，在此过程中，即使出错节点数量增至 300，网络传输中依然没有发生任何包丢弃的现象。

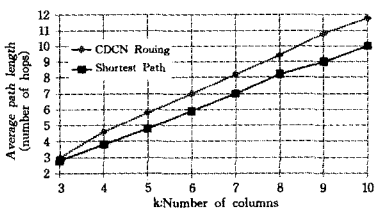

图 4 CDCN 路由过程的平均路径长度

4.3 在容错路由策略中平均路径长度

此外，对容错路由策略也进行了性能仿真测试。实验中，采用(12,4) CDCN 的网络结构。每次测试中，随机选取一定数量的服务器作为出错节点，并随机选择 100,000 对源与目

(下转第 123 页)

函数指针的攻击以及非控制流攻击中对判断相关变量的攻击提出相应的防御方法，借助具有内存染色功能的虚拟机 Valgrind 及其上的工具 Flayer 进行实现，生成针对各种攻击的二进制签名文件，并同时生成二进制补丁文件。使用轻量级检测工具 Pin 进行检测表明，生成的二进制签名文件和补丁文件能高效使用。

参 考 文 献

[1] Vern P. Bro: A System for Detecting Network Intruders in Real-Time[C]//Proceeding USENIX Security Symposium. 1998

[2] Martin R. Snort-Lightweight Intrusion Detection for Networks [C]//Proceeding of LISA '99

[3] Christian K, Jon C. Honeycomb Creating Intrusion Detection Signatures Using Honeypots [C] // Proceeding of the Second Workshop on Hot Topics in Networks. 2003

[4] Hyang-Ah K, Brad K. Autograph: Toward Automated, Distributed Worm Signature Detection[C]//Proceeding of the USENIX Security Conference. 2004

[5] Sumeet S, Cristian E, George V, et al. Automated Worm Fingerprinting[C]//Proceeding of OSDI'04

[6] James N, Brad K, Dawn S. Polygraph: Automatically generating signatures for polymorphic worms [C] // Proceeding of IEEE S&P. 2005

[7] Li Zhi-chun, Manan S, Yan Chen, et al. Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience[C]//Proceeding of IEEE S&P. 2006

[8] Lorenzo C, Andrea L, Luca M, et al. LISABETH: automated content-based signature generator for zero-day polymorphic worms [C] // International Conference on Software Engineering, Proceedings of the fourth international workshop on Software engineering for secure systems. 2008

[9] Aleg K, Wenke L. Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic[R]. Georgia Tech College of Computing, 2004

[10] Roberto P, David D, Wenke L, et al. Misleading Worm Signature Generators Using Deliberate Noise Injection[C]//Proceeding of IEEE S&P. 2006

[11] James N, Brad K, Dawn S. Paragraph: Thwarting Signature Learning by Training Maliciously [C] // Proceeding of RAID. 2006

[12] Ramkumar C, van den Berg E. A fast static analysis approach to detect exploit code inside network flows[C]//RAID. 2005

[13] Christopher K, Engin K, Darren M, et al. Polymorphic worm detection using structural information of executables[C]//RAID. 2005

[14] James N, Down S. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software[C]//NDSS. 2005

[15] James N, David B, Dawn S. Vulnerability-specific execution fltering for exploit prevention on commodity software[C]//Proceedings of NDSS. 2005

[16] Nickolai Z, Hari K, Michael D, et al. Hardware enforcement of application security policies using tagged memory[C]//OSDI. 2008

[17] Liang Zhen-kai, Sekar R. Automatic generation of buffer overflow attack signatures: An approach based on program behavior models[C]//Proceedings of ACSAC. 2005

[18] David B, James N, Dawn S, et al. Towards automatic generation of vulnerability-based signatures[C]//IEEE S&P. 2006

[19] Susanta N, Tzi-cker C. Execution Trace-Driven Automated Attack Signature Generation[C]//ACSAC. 2008

[20] 陈平，韩浩，沈晓斌，等. 基于动静态程序分析的整形漏洞检测工具[J]. 电子学报，2010，38 (8)：1741-1748

(上接第 112 页)

的地址，测试容错路由算法的运行情况。如图 5 所示，出错节点的数量从 1 增至 300，建立平均路径长度与出错节点数量之间的关系。(12,4) CDCN 网络架构中拥有 5,000 台服务器，若其中 300 台出错，其出错概率已达到 6%，网络传输中依然没有发生任何包丢弃的现象。

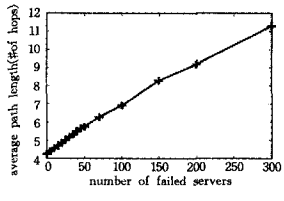

图 5 CDCN 容错路由策略中平均路径长度

参 考 文 献

[1] Amazon elastic compute cloud[EB/OL]. http://aws.amaz-on.com/ec2/

[2] Rabbe L. Powering the Yahoo! network[J]. Nov. 2006

[3] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture[C]//Proceedings of SIGCOMM' 08. 2008

[4] Mysore R N, et al. Portland: A scalable fault-tolerant layer 2 data center network fabric[C]//Proceedings of SIGCOMM' 09. 2009

[5] Cisco data center infrastructure 2.5 design guide[C]//Dec. 2007

[6] Guo C, Wu H, Tan K, et al. Dcell: a scalable and fault-tolerant network structure for data centers[C] // Proceedings of SIGCOMM' 08. 2008

[7] Li D, Guo C, Wu H, et al. FiConn: Using backup port for server interconnection in data centers[C]//Proceedings of INFOCOM' 09. 2009

[8] Guo C, Lu G, Li D, et al. BCube: A high performance, server-centric network architecture for modular data centers[C]//Proceedings of SIGCOMM' 09. 2009

[9] Leighton F T. Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes[M]//M. Kaufmann, 1992

[10] Jeng M, Siegel H. A fault-tolerant multistage interconnection network for multiprocessor systems using dynamic redundancy [C]//Proceedings of ICDCS' 86. 1986