文章编号:1001 - 9081(2007)04 - 0814 - 04

基于频率调节的分布式系统时间同步算法设计与实现

赵 斌[1],贺 鹏[1,2],易 娜[1]
(1. 三峡大学 电气信息学院,湖北 宜昌 443002; 2. 三峡大学 信息技术中心,湖北 宜昌 443002)
(hpeng@ctgu.edu.cn)

摘 要:为了降低 Internet 上对 NTP 时间服务器的访问频率,有效缓解时间服务器资源负担过重的状况,提出了一套适用于分布式系统的基于频率调节的时间同步算法。实验表明,该算法在保障同步精度的前提下,相对于传统的建立在相位调节方式上的时间同步算法,有较好的效果。

关键词:频率调节;分布式系统;时间同步;同步算法
中图分类号: TP393.2;TP301.6 **文献标识码**:A

Design and implementation of time synchronization algorithm based on frequency adjustment in distributed system

ZHAO Bin[1], HE Peng[1,2], YI Na[1]
(1. *College of Electrical Engineering and Information Science, China Three Gorges University, Yichang Hubei* 44300, *China*;
2. *Information Technology Center, China Three Gorges University, Yichang Hubei* 44300, *China*)

Abstract: In order to reduce access frequency to the NTP time server, and efficiently relieve the over-load situation of server. A time synchronization algorithm based on frequency adjustment in distributed system was proposed. Under the same accuracy requirement, the experimental results show that the algorithm is more effective than the traditional algorithm based on the method of phase adjusting time.

Key words: frequency adjustment; distributed system; time synchronization; synchronization algorithm

0 引言

随着计算机日益深入地走进人们的工作和生活,信息技术和网络技术的许多应用都越来越需要高精度、高可靠性的时间同步。目前,NTP 是实现网络时间同步的主要工具,它主要采用锁相环技术对本地时钟进行相位调节[1,2]。通过分析和研究发现,这种方法常常因网上对时间服务器访问量大,从而导致服务器负担过重,效率下降的现象[3,4]。针对这种情况,我们采用艾伦方差分析时间测量过程中的误差,并对不同类型的误差提出相应的优化算法。

艾伦方差是一种时域分析工具,被用来分析时间系列的频谱密度特征,表征时间频率稳定性;噪声的频谱密度可以表明误差的种类和来源[5,6]。本文通过艾伦方差分析时钟、测量过程和网络三个环节误差的来源、类型和统计特性,得到一个相对准确的误差估量,在已有同步算法[7~9]和经典网络时间协议 NTP 研究的基础上[10],构建一套以频率调节为核心的纯软件实现的时间同步控制算法,以满足越来越多的分布式系统应用对高效率、高精度时间同步的需求。

1 算法模型的建立

1.1 艾伦方差描述

艾伦方差通常用 $\sigma_y^2(\tau)$ 表示[11]:

$$\sigma_y^2(\tau) = < \frac{1}{2}(y_{k+1} - y_k)^2 > = \frac{1}{2\tau^2(N-2)} \sum_{k=1}^{N-2} (x_{k+2} - 2x_{k+1} + x_k)^2 \quad (1)$$

测量值 x_k 代表在 t_k 时刻本地时钟和服务器的时间差的测量值。

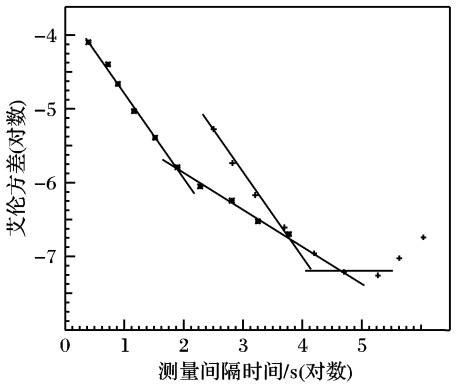

图 1 自由运行时钟艾伦方差

图 1 是一自由运行时钟在两种不同测量方式下的艾伦方差图。“*”是与标准时钟通过总线直接连接测量所得。“+”是该时钟选择 Internet 上标准时钟源作为标准时间测量所得。从图中看出,对于两种不同测量方式,起始角度斜率都是 -1,表明在此段时间内测量过程的相位白噪声占了主导地位。统计量 *S*1 的测量选择在此时间段,可以有效抑制测量过程相位噪声。随着测量间隔的延长,角度斜率变为 -0.5,此时频率白噪声占据主导地位。统计量 *S*2 的测量选择在此区间内。到大约 30 000s 时,也就是 9h 以后,角度斜率变为 0。频率的随机抖动、老化和温度影响逐渐占据主导,对 *S*2 的测量不能超过此时限。另外,当选择 Internet 上标准时间源时,可以看出在相对较长时间内,测量噪声占据主导地位,这是网

收稿日期:2006 - 10 - 17;**修订日期**:2006 - 12 - 15

作者简介:赵斌(1979 -),男,山东滕州人,硕士研究生,主要研究方向:计算机网络和智能信息处理; 贺鹏(1965 -),男,湖北当阳人,教授,主要研究方向:计算机网络和分布式计算机系统; 易娜(1980 -),女,湖北宜昌人,硕士研究生,主要研究方向:计算机网络和智能信息处理.

络噪声的缘故。

1.2　算法描述

根据上述的分析,可以把对相位差的测量时间间隔控制在相位白噪声占优势的时间范围内,对频率偏差的测量时间间隔控制在频率白噪声占优势的时间范围内,采用统计的方法来达到较为准确的测量和估算,建立一个时间同步算法的模型。

为了实现对偏差的分离,在每一次调整周期结束后,都要重新计算下面两个统计量:1)$S1$:时间间隔很小的一组时间偏差测量值的标准差;2)$S2$:时间偏差的预测值与测量值之间的误差。时间差从本质上说是时钟频率偏差造成的:

$$\hat{x}_{k+1} = x_k + \hat{y}_k \Delta t \tag{2}$$

$\hat{x}_{k+1}$ 是表示在 t_{k+1} 时刻时间偏差的预测值,x_k 是 t_k 时刻时间偏差的测量值,$\hat{y}_k$ 是 t_k 时刻频率偏差的预测值,$\Delta t = t_{k+1} - t_k$。预测值与真实值的偏差是:

$$\varepsilon_{k+1} = | x_{k+1} - \hat{x}_{k+1} | \tag{3}$$

由于测量间隔小,硬件时钟的参数保持相对稳定,此时测量过程和网络中的抖动是噪声的主要来源,该噪声属于相位白噪声,因此统计量 $S1$ 表征的是测量过程中的相位白噪声。对于相位白噪声,利用统计方法可以有效地减小测量结果的偏差。具体的说,如果该统计量大于所要求的时间精度,则增加每组测量的个数,反之则减少,直到两者基本相等。该标准值代表着整个算法的精度值,增加适当的测量的个数能提高算法的精度。但精度本质上都是由时钟本身的频率抖动决定的,使用软件的方法只能达到毫秒级的精度。

在算法进入稳定运行后,即所有参数基本调整平衡后,$S2$ 表征的是系统时钟的频率稳定性。如果预测误差的平均值远大于测量噪声,即 $S2 \gg S1$,说明此时时钟修正周期过长,时钟的频率误差主要决定于频率抖动。换句话说,时钟的性能不足以支持这么长时间的频率稳定性,需要减小修正周期。反之则说明时钟频率稳定性能够支持更长的时钟修正周期,此时增加修正周期并不会对算法的性能造成大的影响,却能够降低同步算法的请求频率,提高算法效率。

如果预测误差的平均值与测量噪声一致,则说明当前的时间偏差预测准确有效。我们用这两个统计值更新系统的参数并用频率偏差的估算值来调整系统时钟的频率。

对于频率偏差的估算采用以下两个步骤。

首先计算 t_k 时刻的频率偏差:

$$y_k = (x_k - x_{k-1})/\Delta t \tag{4}$$

然后与之前的频率偏差进行线性平均:

$$\overline{y_k} = (\overline{y_{k-1}} + G y_k)/(1 + G) \tag{5}$$

其中 $G \approx \tau_0 / T_{n\omega}$,$T_{n\omega}$ 由自由振荡晶体的二维阿仑方差曲线斜率从 -0.5 到 0 增加时的校正周期的 τ 值决定。

对于相位差的调整并不是一次调整到标准值,而是采用分阶段逐步调整的方法。因为网络和时钟本身的原因调整过程中会产生微小的波动,一般而言这个值大约为 0.5ms。因此第 K 次校正周期中每次微量调整间隔为:

$$T_k = \frac{0.0005}{\overline{y_k}}$$

如果 $\overline{y_k} = 1.15 \times 10^{-5}$,校正周期 $\tau_0 = 3000$s,为了正好 75 次调整完毕,则调整 $T_k = 40$s。因此每 40s 能调整 460μs,进行到第 75 次调整后进行新一轮的校正循环。

完整算法由两个循环组成:外循环,即每隔 τ 时间对模型的参数进行更新;内循环,即每隔 t_k 对时钟进行微调。

如果时间偏差测量值的标准差在精度范围内,并且偏差预测值和实际值基本一致,那么当前的时间偏差测量值被用来更新算法中的参数。这其中包括更新 $S1$ 和 $S2$ 的平均值与频率偏差的平均值,并且用该值进行调节客户端时钟频率。

2　算法的实现

2.1　算法系统结构及流程

通过上述描述和分析,本文采用了与 NTP 相似的算法结构模型,但抛弃了 NTP 协议为了适用于多种环境而带来的复杂和不实用。该模型具体包括通信、偏差测量、时钟过滤、频率预测和时钟调节五个模块,它们之间的关系如图 2。

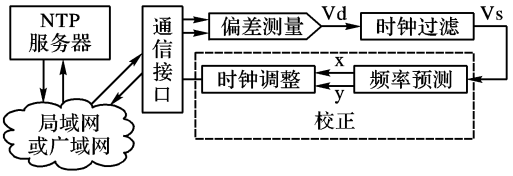

图 2　算法系统结构

其中 Vd 是测量的相位差,Vs 是有效的测量数据,X 是校正后的时钟相位差,Y 是校正后的频率漂移。整个系统是一个负反馈控制系统。总的算法流程见图 3。

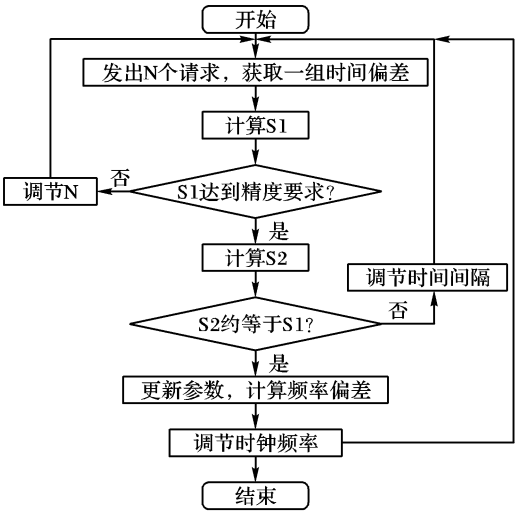

图 3　同步算法流程

2.2　算法的实现

基于对算法可应用性及运行速度的考虑,我们选择了 Windows 2000 与VC++ .net 设计开发平台。

2.2.1　通信模块

在通信模块中采用一个数据结构来填充 SNTP 报文,其代码如下:

```
struct NtpBasicInfo
{
  BYTE m_LiVnMode;
  BYTE m_Stratum;
  char m_Poll;
  char m_Precision;
  long m_RootDelay;
  long m_RootDispersion;
  char m_ReferenceID[4];
  CNtpTimePacket m_ReferenceTimestamp;
  CNtpTimePacket m_OriginateTimestamp;
  CNtpTimePacket m_ReceiveTimestamp;
  CNtpTimePacket m_TransmitTimestamp;
}
```

通信主要步骤：

1）调用 socket(domain,type,protocol)建立套接字。其中 domain = PF_INET(Internet 域);type = SOCK_DGRAM(UDP 数据报);protpcol = IPPROTO_IP(IP 协议);

2）使用 NtpBasicInfo 结构填充一个请求报文,其中 m_LiVnMode = 00 011 011,LI = 00 表示无预告。VN = 011 表示 NTP 第 3 版本。Mode = 011 表示点对点模式。用本机系统时间填充 m_TransmitTimestamp ;

3）调用 sendto(socket_id,buf,buflen,flags,to,tolen)发送 SNTP 客户端请求,其中 buf 为 SNTP 数据,用 NtpBasicInfo 结构填充;

4）调用 recvfrom (socket_id, buf, buflen, flags, from, fromlen)接收 NTP 服务器返回的时间,其中 buf 存储着服务器返回的数据包,结构为 NtpBasicInfo;

5）从返回的数据包中取出有用的时间戳信息,计算延迟和本地时钟偏差,并提供给系统。

2.2.2 偏差测量模块

在时间偏差测量中,为了减小误差提高精度,我们采用了两种方法:1)本地时间并非直接从系统读取,而是从一个高精度的时间源读取时间基点,然后通过性能计数器计算得到,其读取精度可以达到毫秒以下;2)根据测量过程中的噪声特点,用一组时间偏差平均值代替单个时间偏差,降低网络延迟抖动和操作系统硬件噪声带来的测量误差。

性能计数器是一个高精度的硬件计数器,通过在一个紧凑循环内不断重复把性能计数器值和对应的系统时间进行同步,可以提供微秒级的计数。据此能够计算出一个高精度的当前系统时间。

2.2.3 时钟过滤模块

取一组测量值的平均值的方法可降低网络抖动带来的误差,但是在数小时的同步过程中仍然会出现明显不合理的异点。因此在时钟过滤模块中只保留最近的 5 次历史数据作为标准判断当前值是否为异点,如果是则抛弃当前值,并重新向服务器发送一组时间请求。

2.2.4 频率预测模块

根据前述,对于频率偏差的第一次取值采取下面的方法:计算机启动后,每隔 5min 向时间服务器发送一组时间请求,持续 3h,本地时钟保持自由运行。这样得到 36 个时钟偏差值,记录在数据库中。然后对这 36 个值进行线性回归分析,得到时间偏差的斜率,该斜率即为这 3h 内时钟频率偏差的一个估算。有了第一次的取值,在此后的 3 个周期内,用式子(5)对频率偏差进行线性平均,然后调节算法开始运行。

采用简单的最小二乘法进行线性回归,其数学表达:假设直线方程为 $\tilde{y} = ax + b$,误差的平方和 S 为:

$$S = \sum_{i=1}^{m} (y_i - \tilde{y})^2 \tag{6}$$

把 $\tilde{y}$ $a + bx$ 代入上式,得:

$$S = \sum_{i=1}^{m} (y_i - (ax_i + b))^2 \tag{7}$$

如果 S 值越小,则线性拟合方程越准确。为使 S 达到最小值,把上式对 a,b 微分,令它们为零,得到方程组:

$$\begin{aligned} mb + a\sum_{i=1}^{m} x_i &= \sum_{i=1}^{m} y_i \\ b\sum_{i=1}^{m} x_i + a\sum_{i=1}^{m} x_i^2 &= \sum_{i=1}^{m} y_i x_i \end{aligned} \tag{8}$$

即可解得 a,b:

$$b = \frac{1}{m}\sum_{i=1}^{m} y_i - \frac{a}{m}\sum_{i=1}^{m} x_i \tag{9}$$

$$a = \frac{m(\sum_{i=1}^{m} y_i x_i) - (\sum_{i=1}^{m} x_i)(\sum_{i=1}^{m} y_i)}{m(\sum_{i=1}^{m} x_i^2) - (\sum_{i=1}^{m} x_i)^2} \tag{10}$$

通过线性回归计算可得拟合直线的方程是:$y = ax + b$;其中 x 的系数 a 即为频率偏差,作为同步算法的频率偏差初始值。

2.2.5 时钟调整模块

在对本地时钟偏差测量处理完成后,采用调频和调相相结合的方法调整时钟达到与时间服务器的同步。通常当客户机刚刚启动时会有较大的时间偏差,此时采用调相方式一次同步到位,相位偏差的补偿直接使用 API 函数 SetSystemTime 即可实现。当同步算法进入到稳定运行后,时钟会按照它自己的频率运行,逐渐产生频率偏差。算法用 GetSystemTimeAdjustment 获得 tick 值的大小;使用多个标准频率以一定的时间比例调节时钟,让其平均值等于需要补偿的所需精度的频率偏差;采用多媒体定时器 timeSetEvent 获得精度为 1ms 的精确延时;通过 API 函数 SetSystemTimeAdjustment 来调节频率。

另外,在确认系统软件发生了漏掉 tick 中断的情况时,进行相应的相位补偿。

3 模拟仿真及结果分析

为了验证本文算法的有效性和可行性,本文的模拟仿真利用校园局域网,采用 C/S 通信模式进行测试。图 4 是模拟仿真网络拓扑。其中 NTP 时间服务器选择美国的微软公司服务器(www. time. windows. com)或者中国的国家授时中心(www. time. ac. cn)。它们属于 NTP 时间同步网络中的第一级别的服务器,其时间直接与 UTC 时间同步,精度和误差都非常好。本次实验选定美国的微软公司服务器作为算法试验中的 NTP 服务器。

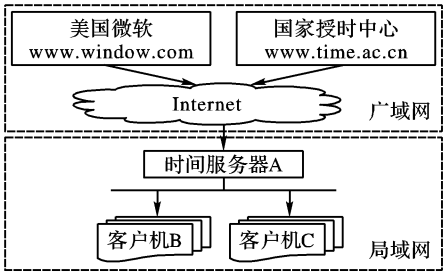

图 4 模拟实验拓扑图

3.1 广域网同步结果

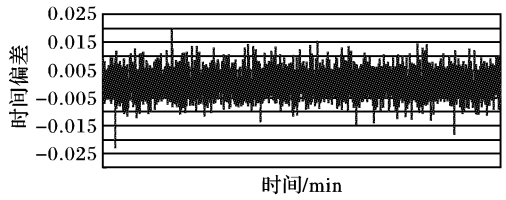

图 5 6 天时间偏差统计

图 5 是算法运行 6 天的客户端和 NTP 服务器之间的时间数据偏差图,每隔 1min 测量一次。经过计算,算法总体标准方差均值为 3. 92ms,最大偏差为 ± 25ms,平均偏差为 3. 32ms。

3.2 局域网同步结果

图 6 是算法运行 24h,同样每隔 1min 测量一次。与广域网相比较,局域网内时间同步时间延迟和抖动相对较小,分别是 0.07665 秒和 0.0069,从而同步的精度和误差优于广域网内时间同步。客户端的时间偏差 < 10ms。经计算,标准方差均值为 2.31ms,最大偏差为 ±15ms,平均偏差为 3.08ms。

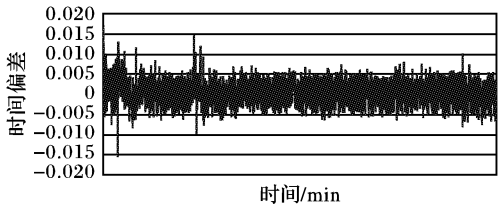

图 6　24 小时时间偏差统计

3.3 结果分析

本文的算法取得了最大偏差小于 20ms,平均绝对偏差小于 5ms,偏差的标准方差均值小于 5ms 的同步精度,表明该算法具有良好的精度和稳定性。在算法效率上,相同精度的普通算法发送时间请求的频率大约是 360 个/h,本算法的平均请求频率约为 70 个/h,算法效率得到很大提高。稳定性方面,相对于传统的调相算法,该算法可以避免把测量中的相位噪声带入时钟,并且其时钟曲线相对更加平滑,不会造成时间值大的抖动,这些都使得以时间为基础的其他应用程序更加稳定,因此,更容易获得较广泛的应用。

4 结语

本算法在对 NTP 核心算法和同步机制分析的基础上,借鉴了传统算法中的 C/S 模式,从精度、效率和实用性这三个方面考虑,构建了一套以频率调节为基础的网络分布式系统中的时间同步算法。通过理论和实验证明,该算法在减少测量误差,调节本地时钟和算法效率这三个方面都取得了良好的效果,可以推广到对时间要求较为严格的分布式网络系统中。今后的工作是在保证算法稳定性的同时,为优化时间偏移量,引入误差侦测机制,以进一步提高算法精确度。

参考文献:

[1] MILLS D. Network Time Protocol Ver3 Specification, Implementation and analysis, RFC 1305[S].

[2] JOHANNESSEN S. Time synchronization in a local area network [J]. Control Systems Magazine, IEEE, 2004, 24(2): 61 – 69.

[3] LEONOV GA, SELEDZHI SM. Programmable phase locked loops for digital signal processors[A]. Proceedings of the 2003 International Conference on Physics and Control[C]. 2003, 2: 548 – 554.

[4] 胡华春, 石玉. 数字锁相环原理与应用[M]. 上海: 科学技术出版社, 1990.

[5] BARNES JA, CHI AR. Characterization of Frequency Stability[J]. IEEE Transactions on Instrumentation and Measurement, 1971, 20(2): 105 – 120.

[6] OMER G, ISRAEL C, MOSHE S. Network time synchronization using clock offset optimization[A]. Proceedings of the 11th IEEE International Conference on Network Protocols[C]. Atlanta, GA, USA, 2003. 212 – 221.

[7] 贺鹏, 李菁, 吴海涛. 网络时间同步算法研究与实现[J]. 计算机应用, 2003, 23(2): 15 – 17.

[8] 贺鹏, 吴海涛. 分布式系统的时间同步算法研究及应用[J]. 计算机应用, 2001, 21(12): 20 – 24.

[9] KITAGUCHI Y, MACHIZAWA A. Research of advanced time synchronous system with network support[A]. Conference on Communications, Computers and Signal Pacific Rim[C]. 2003, 2: 1036 – 1039.

[10] MILLS DL. A brief history of NTP time: Memories of an Internet time – keeper[J]. ACM SIGCOMM Computer Communication Review, 2003, 33(2): 9 – 21.

[11] 黄秉英, , 肖明耀, 马凤鸣. 时间频率的精确测量[M]. 北京: 中国计量出版社, 1986.

(上接第 813 页)

4 结语

聚合组播是一种全新的解决组播规模问题的方法,它的关键在于如何选择聚合组播树,传统的贪婪算法有一定的局限性。本文提出了一种新颖的自适应拉格朗日松弛算法,它能够动态调整拉格朗日乘子,使找到的解尽量接近于全局最优解。仿真实验结果表明,这种算法在提高聚合度、减少转发状态上优于传统的贪婪算法,而且算法执行效率比较高。下一步的工作是用其他现代优化算法如模拟退火算法、遗传算法等来解决聚合组播问题;另外,组播组成员动态加入、离开的动态聚合组播问题也需要考虑;此外,能满足 QoS 要求的聚合组播也是下一步研究的重点。

参考文献:

[1] DEERING S. Multicast routing in a datagram internetwork[D]. Ph. D thesis, 1991.

[2] RADOSLAVOV PI, ESTRIN D, GOVIDAN R. Exploiting the bandwidth-memory tradeoff in multicast state aggregation[R]. Computer Science Department, University of Southern California, 1999.

[3] THALER D, HANDLEY M. On the aggregatability of multicast forwarding state[A]. Proceedings of IEEE INFOCOM[C]. 2000. 26 – 28.

[4] BOIVIE R, FELDMAN N, IMAI Y, *et al.* Explicit multicast (Xcast) basic specification[DB/OL]. http://www.ietf.org/internet-drafts/draft-coms-xcast-basic-spec-06.txt, 2004.

[5] GKANTSIDIS C, MIHAIL M, ZEGURA E. Spectral analysis of Internet topologies[A]. Proceedings of the IEEE INFOCOM[C]. 2003. 364 – 374.

[6] SONG S, ZHANG ZL, CHOI BY, *et al.* Protocol independent multicast group aggregation scheme for the global area multicast[A]. Proceedings of the IEEE Global Internet Symposium[C]. 2000. 259 – 266.

[7] FEI A, CUI J-H, GERLA M, *et al.* Aggregated Multicast: an approach to reduce multicast state[A]. Proceedings of Sixth Global Internet Symposium[C]. 2001. 124 – 126.

[8] FEI AG, CUI JH, GERLA M, *et al.* Aggregated multicast with inter-group tree sharing[A]. Proceedings of NGC2001[C]. 2001. 341 – 345.

[9] CUI JH, KIM JY, MAGGIORINI D, *et al.* Aggregated multicast-A comparative study[A]. Proceedings of IFIP Networking[C]. 2002. 19 – 24.

[10] GAREY MR, JOHNSON DS. Computers and Intractability: A Guide to the Theory of NP-Completeness[M]. SanFrancisco, W. H. Freem and Co, 1979.

[11] KARP RM. Reducibility among combinatorial problems[A]. Complexity of Computer Computations[C]. New York, Plenum Press, 1972. 85 – 103.

[12] BEASLEY JE. A Langrangean heuristic for set-covering problems [M]. Naval Research Logistics, 1990.

[13] AT&T IP Backbone[DB/OL]. http://www.ipservices.att.com/backbone, 2001.