行政院國家科學委員會補助專題研究計畫期中報告

※※※※※※※※※※※※※※※※※※※※※※※※※

※ **基於 64 位元作業系統之高可用度與自我管理** ※

※ **的 Web Services 平台之研究(I)** ※

※ **Research on Highly Available and Self-Managed** ※

※ **Web Service Platform based on 64-bit Operating** ※

※ **Systems(I)** ※

※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑ 個別型計畫　　□ 整合型計畫

計畫編號：NSC　92-2213-E-009-069

執行期間：92 年 8 月 1 日至 93 年 7 月 31 日

計畫主持人：　張瑞川　　國立交通大學資訊科學學系

計畫參與人員：　張大緯　　國立交通大學資訊科學學系

李岳峰　　國立交通大學資訊科學學系

江英杰　　國立交通大學資訊科學學系

蔡德聖　　國立交通大學資訊科學學系

林建豪　　國立交通大學資訊科學學系

吳秉南　　國立交通大學資訊科學學系

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊科學學系

中　華　民　國 九 十 三 年 五 月 十 日

行政院國家科學委員會專題研究計畫期中報告

基於 64 位元作業系統之高可用度與自我管理的 Web Services 平台之研究(I)

Research on Highly Available and Self-Managed Web Service Platform based on 64-bit Operating Systems (I)

計畫編號：NSC 92-2213-E-009-069

執行期限：92 年 8 月 1 日至 93 年 7 月 31 日

計畫主持人： 張瑞川 國立交通大學資訊科學學系

計畫參與人員： 張大緯,李岳峰,江英杰,蔡德聖,林建豪,吳秉南

國立交通大學資訊科學學系

一、中文摘要

現今的伺服器系統需要專業的系統管理者來維護整個系統。隨著軟體系統的日趨複雜，即使是專業的系統管理者也會意外地造成系統失誤。所以一些研究機構開始研究設計下一代可以自我管理(Self-Management)的伺服器系統。其中包括 IBM 的 Autonomic Computing 及 Oceano 計畫，Berkeley 與 Stanford 大學的 Recovery Oriented Computing 計畫，以及 Duke 大學的 Software Rejuvenation 計畫都強調系統在面對某些環境變化（例如：發生錯誤或負載過重）時，能自我做出反應。同時，在這個以客戶為中心的時代，要如何在使用者察覺不到的情況下進行伺服系統的更新及修復將是一個益形重要的課題。

此外，64 位元的平台（主要包含 64 位元的 CPU 與作業系統）將逐漸佔據伺服器市場。然而，64 位元作業系統並非只是 32 位元作業系統的擴充。因為虛擬記憶體空間一旦擴充到 64 位元，整個系統的核心架構必須有新的考量。再者，由於 Web Services 的崛起，未來大型伺服器最廣泛的應用之一便是作為 Web Services 的基礎平台。為了提供不間斷的 Web Service，如何讓 Web Service 的核心（i.e., Web Servers）變成高度可用(Highly Available)便成為一個重要課題。

鑑於上述原因，我們提出一個三年期計畫，目標為建立一個提供 Web Services 的高可用度，自我修復，自我校調的 64 位元作業系統。我們強調這些修復及校調必須在在使用者察覺不到的情況下進行。我們將專注於以下議題：HTTP 要求的轉移，IA64 Linux 通用系統 IO 通道及伺服器代送機制，IA64 Linux 上單一位址空間擴充，元件化並具可部分重新啟動功能之 WWW 伺服器，IA64 Linux 上自我效能偵測系統及自我動態調整系統。藉由此三年計畫，我們將會設計出下一世代，以 64 位元作業系統為基礎，並具有自我修復能力的高可用度 Web Service 平台。

在第一年度，我們完成了 HTTP 要求的轉移與通用系統 IO 通道的設計及實作。本報告將描述這方面的成果。

關鍵詞：64 位元作業系統，IA64，高可用度，自我修復，自我校調，Web 服務

Abstract

A main drawback of the existing servers is that they require system administrators to maintain the operations of

systems. As systems become more and more complicated, fault occurs even thought that the systems are managed by professional administrators. Therefore, researchers start to put efforts on next generation servers, which have the capability of self-management and self-recover. Those projects such as Autonomic Computing and Oceano (hosted by IBM), Recovery Oriented Computing (hosted by Berkeley and Stanford University), and Software Rejuvenation (hosted by Duke University) suggest that systems should react to the changes of the external environment conditions (e.g., fault or overloaded conditions). Meanwhile, since we are in the customer-centric age, how to upgrade/recover systems without disturbing the users is also a critical issue.

64 bit platforms have penetrated the server market gradually. However, 64 bit operating systems are not just the extensions of the 32 bit ones. The design of the whole kernel should have a carefully reconsideration once the virtual address space is extended to 64 bits. In addition, owing to the rise of web services, one of the most common applications of the large server systems is to serve as the platforms of web services. In order to providing continuous web services, it becomes an important issue on making the kernel of the web services (i.e., the web servers) highly available.

Owing to the reasons mentioned above, we propose a three-year project. The goal of the project is to build a highly available, self-healing, and self-tuning 64bit operating system, which provides support to Web Services. We emphasize that the self-healing and self-tuning should be made without user's awareness. We will focus on the following issues: HTTP request migration, Universal IO channel framework on IA64 Linux, server traffic agent, single address space extension on IA64 Linux, component-based and partially-restartable Web servers, self-performance monitoring and self-tuning systems on IA64 Linux. By the three-year project, we will design a next generation, and 64-bit operating system based, web services platform, which is able to healing and tuning itself.

During the first project year, we have completed the design and implementation of the HTTP request migration and the universal IO channel systems, which will be described in this report.

Keywords: 64bit Operating Systems, IA64, High Available, Self-Healing, Self-Tuning, Web Services

二、計畫緣由與目的

電腦系統發展至今，許多研究機構都在思索下一世代的系統應具備何種功能。到目前為止，我們所看的趨勢是，電腦系統將朝兩個方向發展：一是小型的，易於攜帶的嵌入式系統，另一則是大型，具智慧，可自我管理的伺服器系統。前者的技術隨著半導體技術的進步已趨成熟，包含 PDA, Smart Phones, …等等。目前已有多家廠商，包含電信及電腦業者投入其市場。相對而言，後者的研究仍處於萌芽階段。

現今的伺服器系統需要專業的系統管理者來安裝，更新及維護整個系統。隨著軟體系統的日趨複雜，即使是專業的系統管理者也會意外地造成系統失誤。根據 Berkeley 大學的 Recovery Oriented

Computing 計畫的研究，人為造成的系統錯誤佔所有錯誤的 40%。所以一些研究機構開始研究設計下一代可以自我管理(Self-Management)的伺服器系統。其中，最引人注目的研究就是 IBM 的 Autonomic Computing 計畫。IBM 認為下一世代的伺服器系統應該像人體自律神經一樣，具有四種特性：Self-Configuration, Self-Optimization, Self-Protection, Self-Healing。整體來說，系統要能夠根據環境的變化來調整自己，作自我保護及自我修復的工作。而在其先導研究 Oceano 計畫中，系統會根據目前伺服器的負載狀況而自動增減分配給某一服務的伺服器數量。除此之外，Berkeley 與 Stanford 大學的 Recovery Oriented Computing 也提出透過線上自我診斷系統 (On-Line Diagnosis)來提早發現系統潛在的錯誤，自動修復或分析找出錯誤的部分以利管理者修復。在 Duke 大學的 Software Rejuvenation 計畫中，鑑於目前軟體系統大多不夠 Reliable，所以連續執行一段時間後容易發生錯誤。因此，他們提出一個機制使系統會定期 restart 以回到最初，最不易出錯的狀態。這些計畫都強調系統在面對錯誤情況時，要能夠自我修復。然而，二十一世紀會是個以客戶為中心，服務導向的時代。任何系統的更新或錯誤都不應該造成使用者的任何不便。所以要如何在使用者察覺不到的情況下進行伺服系統的更新及修復，將是一個益形重要的課題。

此外，64 位元的平台已經逐漸佔據伺服器市場。作業系統方面，UNIX(包含 Linux)已早有眾多 64 位元的版本(AIX, Solaris, HP-UX, Linux…)，此外微軟的 Windows XP 也有 64 位元的版本。在 CPU 方面，Intel 的 IA64 處理器已經推出第二代 Itanium II，且預計今年會推出 Itanium III。預計未來伺服器系統將以 64 位元為主流。然而，64 位元作業系統並非只是 32 位元作業系統的擴充。因為虛擬記憶體空間一旦擴充到 64 位元，整個系統的核心架構必須有新的考量。例如：作業系統可提供單一位址空間 (Single Address Space)給不同的應用程式，如此一來，便可加快應用程式間溝通(IPC)的效能。或者，一個伺服器軟體程式可以將部分甚至整個檔案系統映對(mapping)到其虛擬記憶體空間。如此一來，原本必須透過高 overhead 的系統呼叫才能達成的檔案系統讀寫便可在直接透過記憶體讀寫完成了。綜言之，64 位元作業系統在設計上必須有新的考量，以期利用其廣大的位址空間來增進系統效能。

再者，大型伺服器未來最廣泛的應用之一便是提供網際網路服務。近來，在網際網路服務的發展上起了一個大變革，那就是 Web Services 的崛起。它定義了一套統一的介面，讓 Web 應用系統間能夠相互溝通。每個 Web 應用系統可提供服務給其他應用系統。如此一來，使得 Web 應用系統的開發得以分工，加速系統開發的時程。可以預見的是，未來網際網路服務上必然遍布著各式各樣的 Web Services，不同 Services 間聯合成一套供需網路，共同合作以提供使用者服務。然而，若 Web Services 一發生意外間斷，則會影響到所有用到此 service 的系統，造成一連串的骨牌效應。就好像某一上游物料短缺，將導致其中下游的產業及消費者受到衝擊一般。為了提供不間斷的 Web Service，如何讓 Web Service 的核心（i.e., web servers）變成高度可用 (Highly Available)便成為一個重要課題。

鑑於上述原因，我們提出一個三年期計畫，目標為建立一個專門提供 Web Services 的高可用度，自我修復，自我校調的 64 位元作業系統。我們將專注於以

下四個議題：

1. HTTP 要求的轉移
2. 通用系統 IO 通道及伺服器代送機制
3. IA64 Linux 上單一位址空間擴充 (Single Address Space Extension)及元件化並具可部分重新起始 (Partially Restart)之 HTTP 伺服器
4. 專為高可用度 Web Services 所設計之 64 位元作業系統

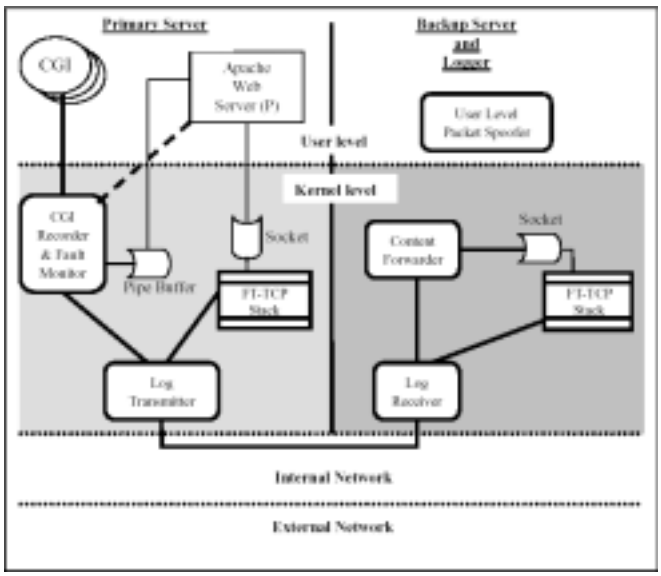
我們將大幅修改 IA64 Linux 內部核心機制, 同時也會發展一個更高可用度的 HTTP server 架構。透過我們所使用的機制，此伺服系統將有希望成為下一世代 Web Services 的平台。

三、結果與討論

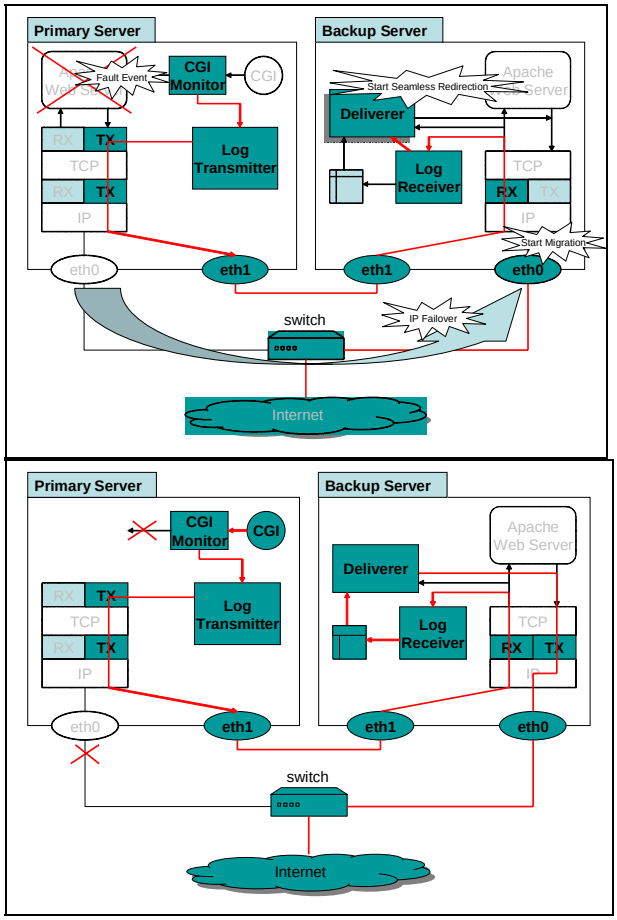
在第一年的計畫中，我們已經完成了 HTTP 要求的轉移及通用系統 IO 通道及伺服器代送機制。

HTTP 要求的轉移：

一般容錯的 HTTP 系統通常是重作 HTTP 要求。這對動態要求（尤其是 transaction-based 的要求）會造成錯誤。所以我們提出一個以要求轉移為基礎的系統。其系統架構如圖一。我們的系統包含一個伺服器叢集. 當一個機器上的 Apache HTTP 伺服程式發生錯誤時，系統會偵測此錯誤並將目前的 HTTP 要求轉移到另一台伺服器上。



圖一、HTTP 要求轉移的系統架構

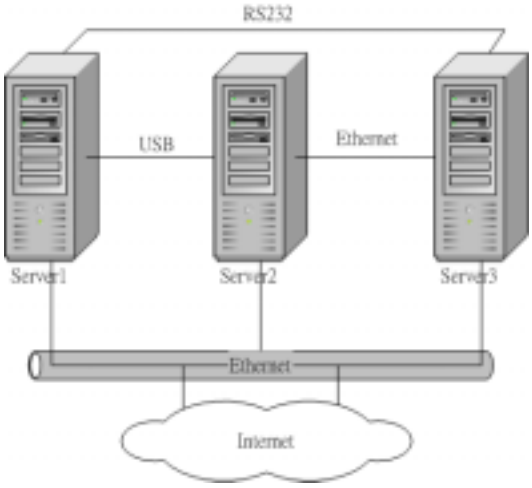


圖二、錯誤偵測與無縫式要求復原

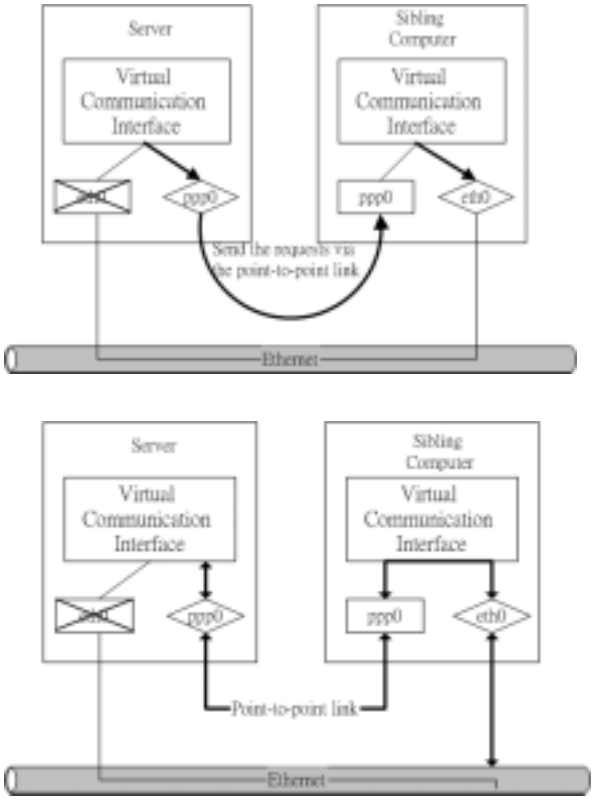
我們的機制包含：動態要求的紀錄，錯誤偵測，無縫式的要求復原…等，如圖二。所有的機制皆實現於 Linux 作業系統核心程式中。

通用系統 IO 通道及伺服器代送機制：

在通用系統 IO 通道及伺服器代送機制方面，我們是從根本上來擴充目前的 redundancy 架構，讓網路出口不再只侷限在網路卡。圖三為一個使用此機制的系統圖。除了一般 Ethernet 外，其它的 IO 通道如：USB，RS232 也可作為傳輸網路資料的媒介。為此，我們在 IA64 Linux 上加入一層通用 IO 通道架構 (Universal IO Channel Framework)，它可以把所有的 IO 通道（包括：RS232，parallel port，USB，Ethernet, Wireless LAN, Bluetooth, IrDA…等等）整合起來並提供一個統一的介面給 IO 通道的使用者。它有兩個好處：其一是隨著伺服器間內部溝通 IO 通道個數的增加，網路服務的 availability 就可以提昇。因為即使一個 WWW 伺服器的外部溝通網路壞掉，它仍可繼續處理客戶端剩下的要求，並將結果透過伺服器間的內部溝通管道先送給另一伺服器，再由此伺服器轉送給客戶端。另外，因為 IO 通道的使用是透過統一的介面，所以 IO 通道的增減並不需要修改使用者的程式碼，如此使得使用者程式碼更易於維護。圖四展示的是一個網路損毀，系統偵測及啟動伺服器代送機制的流程。所有的機制皆實現於 Linux 作業系統核心程式中。



圖三、伺服器代送機制範例系統



圖四、伺服器代送機制流程

四、參考文獻

1. Aghdaie, N., and Tamir, Y., 2001. Client-transparent Fault-tolerant Web Service. In: Proceedings of the IEEE International Conference on Performance, Computing, and Communications, pp. 209-216.
2. Alvisi, L., Bressoud, T. C., El-Khashab, A., Marzullo, K., and Zagorodnov, D., 2001. Wrapping Server-side TCP to Mask Connection Failures. In: Proceedings of INFOCOM 2001, pp. 329-337.

3. Appavoo, J., Hui, K., Soules, C. A. N., Wisniewski, R. W., Silva, D. M. D., Krieger, O., Auslander, M. A., Edelsohn, D. J., Gamsa, B., Ganger, G. R., McKenney, P., Ostrowski, M., Rosenburg, B., Stumm, M., and Xenidis, J., 2003. Enabling Autonomic Behavior in Systems Software with Hot Swapping. IBM Systems Journal, 42(1): 60–76.
4. Barford, P., and Crovella, M. E., 1998. Generating Representative Web Workloads for Network and Server Performance Evaluation. In: Proceedings of the ACM SIGMETRICS '98, pp. 151-160.
5. Brisco, T., 1995. DNS Support for Load Balancing. IETF RFC 1794.
6. Brown, A., and Patterson, D. A., 2003. Undo for Operators: Building an Undoable E-mail Store. In: Proceedings of the 2003 USENIX Annual Technical Conference, pp. 1-14.
7. Candea, G., Cutler, J., Fox, A., Doshi, R., Garg, P., and Gowda, R., 2002. Reducing Recovery Time in a Small Recursively Restartable System. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN-2002), pp. 605-614.
8. Chen, M., Kiciman, E., Fratkin, E., Brewer E., and Fox, A., 2002. Pinpoint: Problem Determination in Large, Dynamic, Internet Services. In: Proceedings of the International Conference on Dependable Systems and Networks (IPDS Track), pp. 595-604.
9. Cristian, L., 1991. Understanding Fault-tolerant Distributed Systems. Communications of the ACM, 34(2):57-78.
10. Davis, T., 2003. Linux Channel Bonding. Available at http://www.sourceforge.net/ projects/bonding/usr/src/linux/Documentation/ networking/bonding.txt.
11. Engler, D., Chelf, B., Chou, A., and Hallem, S., 2000. Checking System Rules Using System-Specific Programmer-Written Compiler Extensions. In: Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI-2000).
12. Garland, M., Grassia, S., Monroe, R., and Puri, S., 1995. Implementing Distributed Server Groups for the World Wide Web. Technical Report CMU-CS-95-144, School of Computer Science, Garnegie Mellon University.
13. Gray, J., and Siewiorek, D. P., 1991. High-Availability Computer Systems. IEEE computer, 24(9):39-48.
14. Horman, S., 2000. Creating Linux Web Farms – Linux High Availability and Scalability. Available at http://www.vergenet.net/linux/has/html/has.html.
15. Intel Corporation, 2003. Intel Networking Technology – Load Balancing. Available at http://www.intel.com/network/connectivity/resources/technologies/load_ balancing.htm.
16. Jann, J., Browning, L. M., and Burugula, R. S., 2003. Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers", IBM Systems Journal, 42(1): 29–37.
17. Kephart, J. O., and Chess, D. M., 2003. The Vision of Autonomic Computing. Computer Journal, 36(1): 41 -50.
18. Luo, M. Y., and Yang, C. S., 2001. Constructing Zero-loss Web Services. In: Proceedings of the IEEE INFOCOM 2001, Vol. 3, pp. 1781-1790.
19. McGrath, R., Kwan, T., and Reed, D., 1995. NCSA's World Wide Web Server: Design and Performance. IEEE Computer, 28(11):68-74.
20. Milz, H., 1998. Linux High Availability HOWTO. Available at
21. http://www.ibiblio.org/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html.
22. Myricom Inc., 2003. Myricom – Creators of Myrinet. Available at http://www.myri.com/.
23. Oppenheimer, D., Ganapathi, A., and Patterson, D. A., 2003. Why Do Internet Services Fail, and What Can be Done about It? In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03). Available at http://roc.cs.berkeley.edu/papers/usits03.pdf.
24. Patterson, D. A., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenheimer, D., Sastry, N., Tetzlaff, W., Traupman, J., and Treuhaft, N., 2002. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. UC Berkeley Computer Science Technical Report UCB//CSD-02-1175.
25. Patterson, D. A., Chen, P., Gibson, G., and Katz, R.H., 1989. Introduction to Redundant Arrays of Inexpensive Disks (RAID). In: Digest of Papers for 34th IEEE Computer Society International Conference (COMPCON Spring '89), pp. 112 -117.
26. Performance Technologies Inc., 2001. The Effects of Network Downtime on Profits and Productivity - A White Paper Analysis on the Importance of Non-stop Networking. White Paper. Available at http://whitepapers.informationweek. com/detail/RES/ 991044232_762.html.
27. Rubini, A., 2000. Making System Calls from Kernel Space. Linux Magazine, Nov. 2000. Available at http://www.linux-mag.com/2000-11/gear_01.html.
28. Scyld Computing Corporation, 2003. Understanding MII Transceiver Status Info. Available at http://www.scyld.com/diag/mii-status.html.
29. Snoeren, A. C., Andersen, D. G., and Balakrishnan H., 2001. Fine-Grained Failover Using Connection Migration. In: Proceedings

of the 3rd USENIX Symposium on Internet Technologies and Systems, pp. 221-232.

30. Stevens, W. R., 1994. TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley Professional, ISBN: 0-201-63346-9.
31. Verma, D. C., Sahu, S., Calo, S., Shaikh, A., Chang, I., and Acharya, A., 2003. SRIRAM: A Scalable Resilient Autonomic Mesh. IBM Systems Journal, 42(1): 19–28.
32. Yang, C. S., and Luo, M. Y., 2000. Realizing Fault Resilience in Web-Server Cluster. In: Proceedings of the IEEE/ACM Supercomputing Conference 2000 (SC2000).
33. Zagorodnov, D., Marzullo, K., Alvisi L., and Bressoud, T., 2003. Engineering Fault-tolerant TCP/IP Services Using FT-TCP. In: Proceedings of the IEEE Dependable Computing and Communications Symposium, pp. 393-402.