網路服務中繼體
A Middleware for Web Services

饒仲華 鄭靜紋 陳明豐
Herman Chung-Hwa Rao Josie Ching-Weng Cheng Ming-Feng Chen

AT&T Lab-Research
iproxy@research.att.com
台北縣板橋市四川路一段 334 號

摘要

iPROXY是一個介於作業系統與網路應用程式間的中繼體(middleware)，支援HTTP通訊協定，採用跨平台的開放式架構。透過iPROXY可共享網路資源、管理快取儲存、進行Web資料處理；iPROXY並提供事件驅動讀取功能，讓伺服器有主動通知客戶端的能力。

關鍵字:中繼體、事件驅動讀取、HTTP

Abstraction

iPROXY is a middleware between OSs and Network Applications with open architecture. Supporting HTTP protocol, iPROXY is transparent to existing browsers and running on different OS platforms. iPROXY allows integrating network services, managing cache, and sharing resources. Event-driven accessing is introduced for client notification.

Keywords: Middleware, HTTP, Event-Driven Accessing, Notification.

1. 緒論

*i*PROXY 是為了發展WWW應用程式而設計的輕巧中繼體(thin middleware)，這些應用程式包括瀏覽器（如Mosaic、Netscape和Microsoft IE）、載入器(loaders)、搜尋引擎(search engines)、指標工具(indexing tools)、資料存取器(robots)、智慧型代理者(intelligent agents)、以及企業網路應用程式(Intranet applications)等。*i*PROXY除了具有取得網際網路服務、分享快取儲存(cache)、及處理Web資料的功能，並提出多種資訊讀取模式，例如事件驅動式讀取(event-driven accessing)、預先讀取、設定預先讀取計劃、以及離線閱讀等。

目前許多瀏覽器，如Netscape, Microsoft IE, 和Mosaic，都提供方便而有效率的方法供使用者搜尋與取得網路資訊。然而大多數的瀏覽器，卻又因為本身包含太多功能(如網路讀取功能、快取管理、Java 虛擬機器、資料顯示、線上搜尋、以及讀取歷史及書籤等)，而顯得負荷過重，甚且有專斷的嫌疑[1]。大部分瀏覽器並未提供開放式架構，不僅瀏覽器之間難以共享快取儲存，而且瀏覽器與應用程式間的交互運作(interoperation)也有困難。其他應用程式更無法使用瀏覽器已發展的功能，以致仍須重複開發某些瀏覽器已有的功能以讀取Web服務。事實上由應用程式複製瀏覽器的特殊功能不僅不合乎效益，也經常顯得困難或甚至無法達成。*i*PROXY藉由中繼體(middleware)的方式解決上述問題。

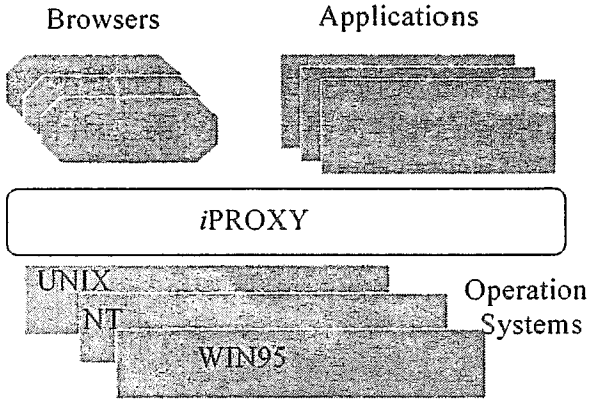

圖1. *i*PROXY—網路服務中繼體

*i*PROXY為Web應用程式的中繼體(middleware)（圖1），提供取得(retrieve)網路資訊及快取(cache)Web資料的應用程式界面(APIs)。對於現存的瀏覽器而言，*i*PROXY的角色就如同一般具有通透性的代理者伺服器(proxy servers)。由於*i*PROXY是以Java和scripting languages例如ksh和perl製作，對於UNIX-like的作業系統，以及微軟Windows 95/NT皆具有可攜性。

*i*PROXY具有下列特點：

- 當使用*i*PROXY為瀏覽器的代理者伺服器，應用程式和瀏覽器可經由*i*PROXY分享快取儲存。因此應用程式以瀏覽器作為圖形使用者界面(GUI)時，可免去重複的網路讀取動作。
- *i*PROXY具有處理HTTP呼叫及Web資料的功

能。例如一個HTTP呼叫可被轉換為不同的HTTP呼叫，或是驅動多個HTTP呼叫。此外，*i*PROXY也可處理來自server端經過壓縮、加密的原始資料，使其恢復為原來的格式再傳回瀏覽器或其他應用程式。

- 藉由延展URL命名方法，應用程式或瀏覽器可直接下達命令給 *i*PROXY。
- *i*PROXY整合HTTP client和HTTP server，將Web的主從(client/server)通訊模式提升為端對端的(peer-to-peer)通訊模式；更把原本由瀏覽器進行資料調查(data polling)的讀取模式加強為由事件驅動(event-driven access)的讀取模式，可藉由本地端時限設定，或透過一般事件驅動Web資料的讀取。

*i*PROXY包含兩個處理(processes)：一個用Java製作的讀取伺服器，以及一個用perl製作的Web伺服器；系統可執行於使用者瀏覽器執行的主機(host)上、當成個人資訊環境(personal information environment)的核心(kernel)，也可作為團體間的系統代理者伺服器(system proxy server)。每個*i*PROXY伺服器都可用其他代理者伺服器作為其代理者伺服器。多個有合作關係的*i*PROXYs可共享快取儲存和網路資源。

本論文其他部份分為六節。第二節介紹*i*PROXY新的特性，包括延展URL (URL extension)、HTML集合(HTML set)、關聯性URL(Associative URL)、以及資料過濾處理(Data filtering)。第三節解釋事件驅動式讀取(event-driven accessing)，第四節敘述*i*PROXY的架構，第五節描述發展於*i*PROXY上的應用程式實例，第六節為本篇論文的結論。

2. 網路服務中繼體(middleware)

*i*PROXY執行於使用者位址空間，並在TCP/IP之上提供HTTP通訊協定[2]，透過IPC (Interprocess communication)與本地端(local)應用程式及遠端(remote)Web servers和proxy servers溝通。其功能基本上可區分為四類：網路功能(networking)，命名功能(naming)，讀取與快取儲存功能(accessing and caching)，以及資料轉換功能(data conversion)。為使中繼體達到提升應用程式效率和加強網路服務的目的，*i*PROXY根據這四類功能提出幾項新特性，包括延展URL、HTML集合、關聯性URL、和Text2URL。本節將概略說明這四類基本功能及這些新的特性。

2.1基本功能

網路功能(networking)

如同一般代理者伺服器，*i*PROXY經由socket連結以接受HTTP呼叫，再與遠端Web伺服器或其他代理者伺服器建立網路連結以取得資料。這項處理的主要工作是選擇路由(routing)：找出到Web伺服器的網路路徑。*i*PROXY根據每個URL位址決定其適當路由，並允許在執行時變更路由。

更進一步，每個*i*PROXY皆可選擇另一個*i*PROXY作為其代理者伺服器[3][4]。為了更有效率地使用*i*PROXY servers之間的通路(channel)，以及提供進一步的功能，我們加強HTTP通信協定，允許重複使用servers間的網路通路，並為其他通訊協定(例如POP3, SMTP,和telnet)提供TCP轉送(forwarding)的服務[5]。

命名功能(naming)

HTTP通訊協定採用URL為基本命名方法[6]，*i*PROXY則藉由URL提供更多元的使用者界面:

a. 延展URL命名方式以包含並傳遞*i*PROXY的命令。
   延展URL的命名功能提供命令模式的界面，應用程式可透過延展URL直接呼叫*i*PROXY的內建功能和命令。
b. 使用別名代替完整的URL (Text2URL功能)。
   將有意義的或甚至任意的非URL字串轉換成對應的或適當的URL，為瀏覽器提供更易懂易記的使用者界面。這部份並支援非英語系國家的文字編碼方式，使用者可用自己熟悉的語言當網址讀取Web資訊。

讀取與快取儲存功能(accessing and caching)

從應用程式的觀點，*i*PROXY就如同一般代理者伺服器，能為各個HTTP呼叫取得Web資料；除此之外，系統還提供多種讀取方式，例如：

a. 預先讀取(future access)：使用內建的排程伺服器執行每日、每星期、或每月的特定資料讀取。
b. 群組讀取(group access)：*i*PROXY系統有能力根據單一的HTTP呼叫而讀取整組相關的Web資料，以達到事先讀取或自動下載(load)資料的功能。為了由單一URL存取多個URLs，*i*PROXY引進HTML集合(set)的觀點—定義首頁及與其相關連的URLs為一個群集(closure)，並以首頁的URL為位址。
c. 事件驅動存取(event-driven access)：除了來自外界呼叫者和內建排程伺服器的HTTP呼叫，*i*PROXY亦可接受外界的通知以驅動Web資料讀取。當事件發生(例如新的Web資料產生)，外界(Web servers 或相關search engines)可藉由*i*PROXY CGI-bins驅動Web資料存取。

資料轉換功能(data converting)

*i*PROXY提供過濾器的窗口以處理來自Web伺服器的原始資料。*i*PROXY把原始資料作為過濾器的輸

入，並將處理過的資料輸出給呼叫者。透過此項功能，*i*PROXY可提供對原始資料進行取代、刪除、或附加資訊的服務。

2.2新特性介紹

延展URL (URL extension)

*i*PROXY藉由延展URL的命名方式掌握中間體功能，並控制個別的Web資料讀取。延展的URL除了包含呼叫者要求的URL之外，也包括處理此呼叫的命令，格式如下：

Requested-URL?iproxy&commandline

其中*Requested_URL*為使用者想要的網頁位址，這部份也可能包含"?"符號。*i*PROXY將"*?iproxy*"視為關鍵字(keyword)處理，其餘部份則被視為命令及參數。例如下列的URL:

http://www.att.com/?iproxy&CacheOff

包含一個呼叫者要求的URL "*http://www.att.com/*"以及一個用以關閉快取機制的命令*CacheOff*; 存取此URL時，*i*PROXY將關閉快取機制，不做Web資料的快取儲存。

命令部份通常是中間體所需的一般功能[8]，如中斷或重新建立網路連結、重建文字與URL(Text-to-URL)間的對應、啟動或關閉Web資料的快取機制、以及存取相關URLs。這些命令將在接下來的章節詳細介紹。

延展的URL容許將命令嵌入HTTP通訊協定中，免於額外再使用一個新的URL; 此外，使用者可用一般瀏覽器發出*i*PROXY的命令，網站發展者也可以將命令嵌入HTML檔案的超連結參考中(hyper reference)；延展URL為應用程式提供一個直接取得*i*PROXY服務與功能的界面。

HTML集合(HTML set)

HTML集合由一些可從單一HTML檔案(稱為根源—root)經由直接或間接方式讀取到的HTML檔案和影像檔案組成，每個集合以其根源網頁的URL命名，並以參數表示整個群集，包括連結參考的深度、是否包括相關影像檔案、以及是否限制超連結參考的範圍。

由URL延展可以設定HTML集合，例如：

http://www.att.com/?iproxy&htwalk=2,local,image

表示一個根源為*http://www.att.com/*的HTML集合，超連結參考的深度為2，只讀取本地端超連結參考，讀取時一併取得所參考到的影像檔案。因此，HTML集合能以單一URL為一組相關網頁命名，指定一個HTML群集(closure)。

HTML集合的特性對於需要下載一系列網頁，如搜尋、索引、預先讀取、或建立mirror等有很大的助益。

關聯性URL (Associative URL)

若讀取URL A時有很大的機率會讀取到URL B，則稱URL B是URL A的關聯性URL。*i*PROXY讀取特定URL時會預先讀取其相關聯的URLs以縮短等待的時間。

關聯性URLs可透過延展URL指定，例如下列的URL：

http://www.att.com/?iproxy&aurl=http://www.att.com/worldnet

指定*http://www.att.com/worldnet*為*http://www.att.com*的關聯性URL。此外，系統建立關聯性URL的對應表，並允許在執行期間更改此對應表。

這項新特性允許加入各種預先讀取的機制。舉例而言，一個預先讀取的機制從*i*PROXY以往的快取儲存中發現某網頁與其所包含影像檔案間的關係，並將這個關係的資訊放入到關聯性URL對應表; 自此每當讀取此網頁時，*i*PROXY將嘗試一併取得與它相關的影像檔案。

資料過濾器 (Data filter)

過濾器以CGI-bin的形式放置在*i*PROXY Web伺服器中，並由*i*PROXY Web伺服器執行。*i*PROXY系統維護URL-pattern與過濾器功能的對應表。若Web伺服器發現取得的資料符合預設的URL-pattern，*i*PROXY系統就會呼叫對應的過濾器，並以Web server取得的資料做為過濾器的輸入; 過濾器的輸出則被傳回給呼叫者。

使用過濾器可達到刪除資料、復原資料、以及附加資訊等功能。來自伺服器端的原始資料可能經過萃取(condensed)、壓縮、加密、或增補(patch)，過濾器將其還原為原來的資料格式，再將資料傳回呼叫者。把提供特定服務功能的過濾器安裝在使用者端，服務提供者可提供特殊資源的服務、允許使用者指定個人喜好、減少龐大的資料下載。

例如，下列為URL-pattern與過濾器對應表中的一項：

*http://foo.com/*secure.html* *http://localhost/bin/decode*

此項目意義為在"*foo.com*"範圍下，所有尾端為"*secure.html*"的URL皆需經過有解碼功能的過濾器"*http://localhost/bin/decode*"處理。

Text2URL

URL命名以ASCII字元表達機關名稱的縮寫、與內容相關的簡寫，甚至可能是不具特別意義的字串; 對非英語系國家使用者而言，URL或許顯得晦澀或冗長。

*i*PROXY提供字串與URL的對應功能，將使用者輸入的字串轉換成對應的URL。*i*PROXY並可接受非ASCII的字串;不同語系使用者可選擇其所使用的語言系統。

當字串恰可對應至單一URL時，*i*PROXY會直接讀取此URL的資料；若比對結果發現有多個URLs與之對應，則*i*PROXY會根據預設的加權函式依序列出所有可能的URLs供使用者選擇，並記憶使用者的選擇結果，作為下次對應時的加權參考。*i*PROXY提供使用者直接編輯此對應表調整加權值的界面。

例如，中央研究院的URL為

http://www.sinica.edu.tw

當使用者由瀏覽器輸入"中研院"時，*i*PROXY自動將其轉換為URL "*http://www.sinica.edu.tw*"，並直接取得此URL對應的資料。

若使用者輸入的字串不在對應表上，則*i*PROXY將呼叫網路上的搜尋引擎，並根據搜尋結果自動增加或修改對應表，使用者也可透過系統界面選擇或增加搜尋引擎。

3.事件驅動式讀取 (Event-driven accessing)

*i*PROXY提供一個內建的Web伺服器，不僅可發出讀取Web伺服器的HTTP呼叫，也執行本地端應用程式的HTTP呼叫。現存的環境大部分只能由瀏覽器向Web伺服器讀取網頁；使用*i*PROXY，遠端的Web伺服器能夠向瀏覽器和使用者發出如網頁更新、新資訊等通知(notification)。這種由Web伺服器通知瀏覽器和使用者進行資料讀取的方法稱為「事件驅動式讀取」。

PointCast 介紹PUSH機制，使伺服器端可將資料傳送(push)到客戶(client)端，取代只靠客戶端查詢或讀取Web資料。然而其所謂的push行為事實上靠客戶端的timeout來驅動資料讀取。*i*PROXY的通知功能則是真正由伺服器端直接驅動push動作。

*i*PROXY的通知機制由CGI-bin **callback**執行。**callback**使用形式為：

/bin/callback?url=URL&msg=message

對每個呼叫，系統會根據呼叫者的IP位址執行下列動作之一：

- 拒絕呼叫並傳回錯誤訊息；
- 記錄呼叫；
- 取得並儲存對應URL的資料；
- 驅動本地端的瀏覽器取得URL資料。

例如：假設使用者Bob在"foo.com"上執行*i*PROXY，一個HTTP呼叫：

http://foo.com/bin/callback?url=http://bar.com/doc/release ?msg="New Release"

通知Bob可取得文件"http://bar.com/doc/release"，執行於Bob機器上的*i*PROXY接收到通知並執行/bin/callback。callback根據Bob指定的回應模式，驅使瀏覽器取得文件、或快取儲存這份文件、或記錄此訊息作為未來參考、或是拒絕這個呼叫。

4. 架構

*i*PROXY包含一個讀取伺服器和一個Web伺服器。*i*PROXY和應用程式(包括瀏覽器)可在同一主機上或在不同主機上，一個*i*PROXY可用其他*i*PROXYs作為網際網路存取時的閘道器(gateway)。

讀取伺服器(Access Server)

讀取伺服器如同一般代理者伺服器，接收來自TCP通訊埠(port)(預設值為8000，可重新組構)的HTTP呼叫，並將其繞道送往對應的Web伺服器、或其他的*i*PROXYs、或是系統代理者(system proxies)。讀取伺服器將取得的資料轉送回呼叫者，並將取得的資料快取儲存在local端磁碟供未來存取之用[9][10]。

即使在未與網路連結的情況，系統仍可提供使用者瀏覽Web資訊的功能。連線中斷的模式下，存取伺服器將以快取資料回應呼叫者，並記錄沒有快取資料的呼叫，當網路恢復連線時，則自動前往取得資料。

IP路由(routing)方面，*i*PROXY透過路由表(routing table)控制路由選擇。路由表描述URL範圍與對應的閘道器；可能的閘道器包括其他*i*PROXYs、系統代理者、或資料所在的Web伺服器。*i*PROXY使用簡單的正規表示式(regular expression)描述URL範圍。路由表的例子如下：

URL範圍	閘道器(Gatway)
http://.research.att.com/**	-
http://.att.com/**	*radish.research.att.com:8000*
http://.tw/**	*www.nctu.edu.tw:8080*
*	*135.104.20.1:80, proxy.att.com:8000*

閘道器若為"-"表示不透過代理者伺服器，而直接向Web伺服器讀取；一個URL範圍可指定多個閘道器。存取伺服器嘗試與表中每一個代理者伺服器聯繫，若與其中任一個代理者伺服器連結成功，則透過此代理者伺服器讀取Web伺服器。如果所處的環境中存在多個系統代理者伺服器，基於網路的安全和效率，*i*PROXY可指定每個系統代理者伺服器負責某些特定區域，例如一個負責本地端、另一個負責內部企業網路、第三個負責公共網際網路等。

Web伺服器(Web Server)

*i*PROXY提供一個支援HTTP通訊協定的內建Web伺服器。與一般Web伺服器的差異在*i*PROXY Web伺服器設計重點在執行功能(functions)，而不只是提供資訊。處理Web資料的過濾器、提供*i*PROXY特

性的功能、甚至其他服務都使用CGI-bin，並以URL命名。
目前*i*PROXY的Web伺服器以perl製作，可執行由描述語言—如perl，ksh—製作的CGI-bins，以及一般的可執行程式(executable codes)。

5. 應用程式

本節介紹幾項利用*i*PROXY特性所開發的應用程式。

載入器 (Loader)

載入器下載HTML集合以供離線閱讀(即使未與網際網路連線時仍可閱讀)。使用者可排定下載網頁的程序，指定每天、每星期、或每個月的資料下載時間，而後在任何時間地點皆可閱讀網站資料。

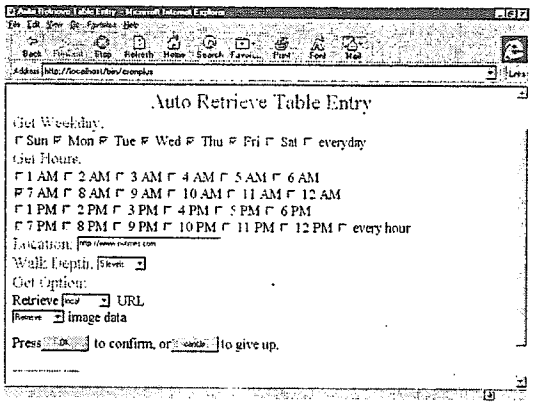

圖2. 排定存取計劃

TOGO

TOGO將HTML集合壓縮並包裝成可容於磁碟或郵寄給其他使用者的單一檔案，也能將此類檔案解開並存放在*i*PROXY的快取儲存中。

使用者定義的附加選單項目(User-defined Menu bar)

*i*PROXY允許使用者將自己定義的選單項目附加於由網際網路取得的網頁下方。選單項目需遵循HTML格式並定義在本地端的檔案中。此項功能支援*i*PROXY內建服務與管理功能捷徑的使用。

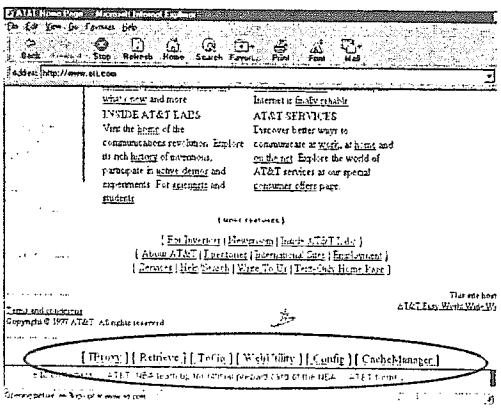

圖3. 使用者自訂附加選單

預先讀取 (Pre-fetching)

*i*PROXY維護一份關聯性URL的列表，每當讀取此表所列的根源URL時，*i*PROXY即會預先取得其相關聯的URL集合。因此，*i*PROXY允許同步取得一個網頁的相關影像檔案，或是根據設定預先取得與此URL相關的網頁。

離線閱讀 (Off-line Reading)

未與網際網路連結時，*i*PROXY以本身的快取儲存回應HTTP呼叫，並記錄無法連結的呼叫；網路恢復連結時，*i*PROXY將自動前往讀取失誤資料。

主動通知 (Notification)

以往Web伺服器只能等待客戶端來讀取資料，無法主動告知客戶端關於資料更新、新增資訊等訊息。主動通知可視為一種事件驅動讀取的應用，由Web伺服器將訊息主動通知客戶端，對於及時性訊息傳佈而言，可避免客戶端不斷對web伺服器端做資料調查(data polling)。

6. 結論

*i*PROXY整合代理者伺服器和web伺服器，以開放架構的中繼體形式執行於作業系統與應用程式之間，提供應用程式界面使網路應用程式和服務能直接使用*i*PROXY的內建功能，方便其發展製作，達到網路資源分享及提升網路服務品質的目的。
*i*PROXY提出的事件驅動式讀取的Web資訊讀取模式，在原有的單向資料查詢模式外又提供雙向的通知和讀取模式；並增加Text2URL簡化傳統URL使用方式。
*i*PROXY仍在繼續發展，未來將加強系統與資料安全方面的考量，並增加效能評估的功能。

參考文獻

1. Butler W. Lampson, Hints for Computer System Design, *IEEE*, pages 11-28, January 1984.
2. T. Berners-Lee, R. Fielding, and H. Frystyk, Hypertext Transfer Protocol – HTTP/1.0. INTERNET-DRAFT draft-ietf-http-v10-spec-0.5.ps, *Internet Engineering Task Force (IETF)*, February 1996. See also at *http://www.w3.org/pub/WWW/Protocols/HTTP/1.0/spec.ps.*
3. Sun Wu, C. C. Liao, Virtual Proxy Servers for WWW and Intelligent Agents on the Internet, *TANet'96.*

4. Harvest A Scalable, Customizable Discovery and Access System, Mar 12, 1995.
5. Transmission Control Protocol, September 1981.
6. T. Berners-Lee, L. Masinter, M. McCahill, Uniform Resource Locators (URL), *http://www.w3.org/put/WWW/Addressing/rfc1738.txt*, December 1994.
7. Peter B. Danzig, Richard S. Hall, Michael F. Schwartz, A Case for Caching File Objects Inside Internetworks, *Technical Report CU-CS-642-93, University of Colorado at Boulder*, March 1993.
8. Philip A. Bernstein, Middleware An Architecture for Distributed System Services, *Digital Equipment Corporation Cambridge Research Lab*, March 2, 1993.
9. Thomas Ball, Fred Douglis, An Internet Difference Engine and Its Applications, *Proceedings of 1996 COMPCON*, pages 71-76, February 1996.
10. Yih-Farn Chen, Glenn S. Fowler, Eleftherios Koutsofios, and Ryan S. Wallach, Ciao: A Graphical Navigator for Software and Document Repositories, *International Conference on Software Maintenance*, pages 66-75, 1995. See also at *http://www.research.att.com/~chen/ciao.*
11. Darren R. Hardy, Michael F. Schwartz, Customized Information Extraction as a Basic for Resource Discovery, *University of Colorado Boulder, Colorado 80309-0430*, March 1994.
12. William G. Camargo, The Harvest Broker, *The Pennsylvania State University, The Graduate School, The Department of Computer Science*, December 1994.