重疊式網路資源監測系統
An Overlay Network Resource Monitor System

*王志誠、劉家佑、陳雍穆、翁政豪、陳昱潭、**鍾添曜
*swswggg@netlab.cse.yzu.edu.tw、**csdchung@saturn.yzu.edu.tw
元智大學資訊工程學系

中文摘要

本研究在對等式網路(P2P)的環境中，設計一種模組化、並以 XML 文件為資料交換媒介的監測系統。本系統建置在一個以地域導向叢集化的自我穩定對等式網路架構—SSIF 之上。SSIF 為可自我調整、穩定(self-stable)、叢集式網路架構，能有效地分散資源管理所需的開支(overhead)。監測代理器(Monitor Agent)的模組化架構，使系統能夠線上(on-line)擴充監測功能。監測資料格式使用 XML 技術描述，易於資料搜索系統的查尋(search)與展示(representation)。此外，監測代理器除了提供即時的監測資料外，也可提供週期性的統計資料。網路管理系統可依據監測代理器提供的資料，提供系統負載平衡、錯誤恢復的管理策略與機制。

中文關鍵字：代理器、整合、分散式資源管理、監測系統。

英文摘要

This study proposes an overlay resource monitor system which is highly scalable, extendable and self-configurable called ORMAN. The monitor system is build upon a SSIF P2P system and is consisted of a monitor agent. The monitor agent contains a set of sensors, called Information Collection Modules (ICMs). ICMs collect the states of monitored attributes, record the attributes and compute the statistical values of monitored attributes in each node. ORMAN provides several resource management mechanisms, such as event notification, admission control, fault tolerance and load balance. The monitor system also provides a set of APIs for developers to design customized ICMs and plug them into system on the fly.

英文關鍵字：Agent，Aggregation，Distributed Resource Monitoring，Monitoring System。

*本研究由國科會計劃，主動式可程式化通訊閘道器之設計與製作(I) 所支持。計畫編號 NSC 93-2213-E-155-044。

1. 導論

近年來積體電路技術的革新、網路的普及化，造就對等式(Peer-to-Peer)[1]-[4]網路架構與網格計算(Grid Computing)的研究和發展。在這些分散式系統中，如何管理(Management)及監測(Monitoring)為數眾多的分散式資源(Storage、Bandwidth、CPU)，成為重要的研究課題[5]-[12]。在對等式網路中，拓樸管理系統藉由網路拓樸中每個節點的資源使用情況，來維持虛擬拓樸的整體效率。在網格計算方面，網格系統佈署應用程式或計算資料時，也需要知道系統的資源使用狀態，用以分配資源(Resource Allocation)，並指派計算工作到計算節點中。資源監測系統，監測並提供網路拓樸中各節點之資源狀態資料，對計算系統與網路系統的資源管理，和服務佈署有極高的價值。

監測系統的架構，由三大部分組成：

A. 網路元件(Network Element；NE)：任何可使用通訊協定跟其他 NE 或網路管理者作資料交換的元件，即為被監測的拓樸節點。在監測系統中，NE 執行一個專門收集系統資源使用狀態，和負責傳送監測資料的軟體。此類軟體通常稱為監測代理器(Monitor Agent)。

B. 網路管理者(Network Manager；NM)：專門收集從 NE 取得的監測資料，並負責統計，及以圖形或表格的方式，顯示所監控 NE 的現況給使用者。在一個監測系統中，可能有一個或多個 NM 同時存在。一個 NE 也可以配有多個 NM。

C. 通訊協定(Communication Protocol；CP)：通訊協定定義 NE 與 NM 之間及相同性質拓樸節點(NE 對 NE、NM 對 NM)溝通方式。

在傳統的主從式網路架構下，SNMP[13]標準規範 NE 與 NM 的溝通方式，輔以 MIB (Management Information Base)[13]，成為網路監測系統的通訊標準。MIB 定義了 NE 負責維護的管理資訊基礎，以及其所內含的資料變數。這些資料變數是 NE 可提供查詢，或更改的系統資源狀態資料。但是傳統的 SNMP 資源管理系統使用中央集權化的架構，這樣的架構在節點數較多的網路中，會產生中央管理的

節點負載過重的問題。

本論文提出一個建置在以地域導向叢集化對等式網路架構上的監測系統。地域導向叢集化對等式網路架構提供穩固(robust)的網路拓樸結構。當被監測的節點數量增加時，叢集系統調整並維持自己的網路拓樸結構，以減少管理的成本，和增加搜尋資料的效率。而可靠性及即時性的監測代理器能提供正確無誤的監測資料。

本研究同時設計一個可程式化的監測代理器，此監測代理器可以使得 NE 提供最新即時的監測資料，與週期性的統計資料，如平均值等。而舊有的監測資料也會儲存在本地資料庫之中，以供日後做歷史性資訊查詢。監測代理器以模組化的方式實做，針對不同的系統資源類型，監測代理器提供相對應的模組使用。使用者也可以自行撰寫模組，進行線上佈屬與監測，使得監測代理器更有彈性(Flexible)。

本文章節如以下所示：第二章將介紹各種的監測系統，第三章描述監測系統本論文所使用的網路架構，和介紹監測代理器系統架構。第四章詳細描述監測代理器的實作形式，第五章總結本文所提出的監測代理器，並且探討未來工作。

2. 相關研究

本章節針對各種不同的監測系統作一介紹：

2.1 SNMP

SNMP(Simple Network Management Protocol)為目前網際網路的網路管理標準，主要由兩種管理實體(Management Entities)─網路管理站(Network Management Station)和 SNMP 代理器(SNMP Agent)所構成。

SNMP 代理器同時也維護稱為管理資訊庫(Management Information Base)的資料結構，此資料結構維持各種可供查詢及更改的本地端系統監測資料，並提供網路管理站以 Get、Get-Next 與 Set 指令對監測資料的進行獲取和設定的操作。

以 SNMP 為核心的傳統式監測系統，在傳送大量的資料時效率仍不及串流式容積傳輸(Stream-based Bulk Transfer，如 TCP。)，採用中央控管的架構也限制整體系統的擴充性(Scalability)。而且 SNMP 的傳輸格式並不適於異質系統間的資訊交換，同時監測資料的呈現方式並無統一的規定，以至於增加監測系統的維護時間和成本。

2.2 SNMP with XML

SNMP 發展至今，仍有需要加強的地方。如監測資料的處理與儲存方式並未有標準格式。傳輸大量的資料時，效率仍不及 TCP 等串流式容積傳輸。SNMP 的設定配置管理(Configuration Management)也未標準化。同時，SNMP 的低階科技 (low-level technology) 特性也使得開發應用軟體的困難度大增。

針對這些缺點，T. KLIE[14]將 SNMP Agent 與 XML 技術結合，以 SNMP-TO-XML Gateway(閘道器)作為解決方案。當網路管理者需要收集監測資料時，將指令轉換成 XML 文件，並傳送至閘道器，由閘道器統一對各個 SNMP Agent 進行資料收集的動作。然而這種方式應用於大型網路系統時，會面對網路流量過於集中在閘道器，使網路負載不平衡(Load Unbalance)，並對監測系統造成單一節點失敗(Single Point Failure)的問題。

2.3 Astrolabe

Astrolabe[9] 使用非結構化 (unstructured) 的 gossip 通訊協定維持一個多階層的樹狀結構，同時提供可動態安裝的 SQL 程式，以便在樹狀結構上進行資料整合和資料查詢的操作，且維持資料的一致性。Astrolabe 的資料整合策略是將子樹中各個節點擁有的資料散佈到子樹中的所有節點；當有一個節點的資料變動時，Astrolabe 必須更新其他擁有相同資料的節點。但是 Astrolabe 並不適用於擁有高變動率的監測資料。而且當 Astrolabe 所維持的監測資料的數量增加時，將會增加網路的資料流通量(throughput)。

在對等式網路中，節點大量的加入與離開，會讓資源管理系統的拓樸不穩定且無法運作。針對此問題，本研究發展一套網路監測系統，能自行處理網路中各種的動態錯誤(dynamic faults)，如節點的不正常離開，並且能在短時間之內將系統整體恢復正常狀態。

同時，ORMAN 使用模組化的監測代理器加強監測功能的擴充性。使用者可線上改變監測代理器的監測功能。監測代理器的遠端部署功能，可針對不同的網路服務進行個別的監測。

3. 監測系統

本論文所提出一套在 P2P 網路環境中的網路資源監測系統，其功能包含：

A. 自我管理(Self-management)：當網路因為節點的加入或離開而需要調整時，監測系統將自行維持網路拓樸。
B. 資料整合(Aggregation)：監測系統把監測資料作有系統地更新、整合[15]與快取(Cache)，並且使用演算法加快查詢速度和提高準確率。
C. 動態佈建監控模組 (Dynamic Deployed Monitoring Module)：使用者可利用監測系統

將監測模組(收集監測資料的模組)從遠端部署至特定的節點中，執行監控動作。在 ORMAN 中，我們使用了模組化的監測代理器達成這一點。

監測系統以一套自我穩定基礎建設式(Self-stabilizing Infrastructure-based；SSIF)的對等式網路作為基礎網路拓樸結構，管理加入監測系統的端點(peer)。本章第一部份介紹 SSIF 的網路拓樸結構，第二部分簡介端點加入和離開 SSIF 的方式，第三部分介紹監測代理器系統架構。

3.1 SSIF 網路拓樸結構

SSIF 為一個三階層結構(Three-level hierarchical structure)、穩定(stable)、區域導向(location-aware)、叢集式(cluster-based)的對等式網路架構，SSIF 有效地分散資源管理所需的開支，如圖一所示，SSIF 網路拓樸由四種端點(peer)構成：

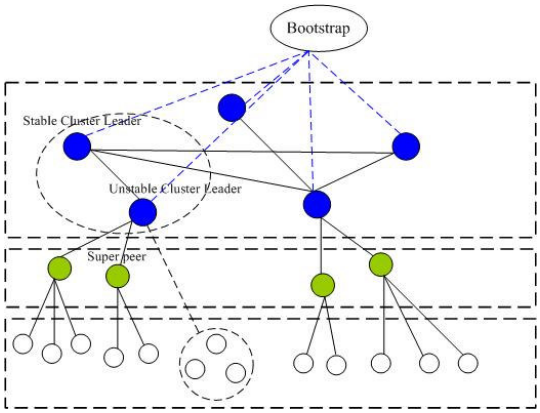
A. 一般端點(Regular peer)：加入 SSIF 的一般節點，並不負責網路拓樸的維持，一般端點同時兼具有兩種功能。一般端點以用戶端(client)的身分送出請求(request)要求超級端點(super peer)、或叢集端點(cluster header)提供服務和查詢資料。同時，一般端點以服務端(server)的身分，提供服務給其他一般端點，如下載檔案或使用儲存空間與中央處理器資源。

B. 超級端點(super peer)：為 SSIF 中，管理一般端點的中間管理者。每個超級節點皆擁用獨立的廣播領域(Broadcasting domain)作為與其直轄的一般端點溝通的管道。超級端點協助廣播領域中的端點發佈(publish)、轉送(forward)資訊，並且處理廣播領域中所有一般端點的加入(join)和離開(leave)的操作。

C. 叢集領導者(Cluster leader)：為網路拓樸的主要管理者。SSIF 以區域導向(location-aware)為原則，將地域相近(close geographically)的端點統合成一叢集，並以能力最強的端點為叢集領導者。叢集領導者為新加入 SSIF 的端點提供定位(positioning)資訊，決定新端點應加入哪個叢集之中，並且提供資訊搜尋的功能。此外叢集領導者與其他的叢集領導者之間，可建立邦聯關係(confederation relationship)。擁有邦聯關係的叢集群，分享彼此的端點資訊進而能提供高效率的全域搜尋(global searching)服務。

D. 起始聯繫(Bootstrap)：起始聯繫提供註冊，及維持和追蹤所有叢集領導者的狀態；當一新叢集被建立時，其叢集領導者需向起始聯繫註冊。起始聯繫也幫助新加入的端點找尋適合加入的叢集，維持 SSIF 區域導向的特性。

SSIF 以叢集為管理單位，每個叢集只有一個叢集領導者，叢集領導者管理複數的超級端點，每個超級端點又管理複數的一般端點。



圖一、SSIF 網路拓樸結構。

3.2 端點加入和離開 SSIF 的方式

加入 SSIF 的端點不需事先設定，而是在端點加入時，由 SSIF 分配至不同的叢集中；端點離開叢集時，SSIF 也會自動調整拓樸結構。

在叢集中，超級端點和叢集領導者擁有維持網路拓樸和其他端點狀態的重要資訊，因此，SSIF 中每個超級端點和叢集領導者都有一個備選端點(Candidate peer)；備選端點必須和叢集領導者(或超級端點)周期性地同步所有的 SSIF 拓樸資訊，當叢集領導者(或超級端點)離開 SSIF 時，備選端點能快速替代原有端點，維持 SSIF 拓樸的完整性。

SSIF 對每個叢集端點的數量有一定的限制；端點數量稀少的叢集將被漸進式的合併(migrating)至其他叢集中，端點數量過多的叢集將被分割(split)成數個小叢集。目前 SSIF 的實作中，一個叢集最大端點容納數為 256 個。

一個端點加入 SSIF 時，皆是以一般端點的身分加入，當同叢集中的叢集領導者或超級端點離開時，能力優秀的一般端點將有機會成為叢集領導者或超級端點。端點加入 SSIF 的動作分成兩階段；首先，新端點在區域網路(LAN)以廣播的方式詢問是否存在已加入 SSIF 的端點。如果有已加入端點，則藉由已加入的端點，進行加入的步驟。不論已加入端點是否為超級節點，新端點加入與已加入端點相同的廣播領域，屬於一個超級節點的管轄。超級端點儲存新端點的資訊，並且向叢集領導者發出有新端點加入的訊息。

如區域網路中沒有任何的 SSIF 端點存在，新端點將直接跟起始聯繫作註冊的動作，要求加入 SSIF。這時，起始聯繫將送出網路中所有叢集領導者的列表，由新端點計算與各個叢集領導者之間的相對位置，來決定新端點應該加入的叢集。位置的測量由新端點、起始聯繫、和另外兩個隨機挑選的叢集領導者之間的距離(ping)來決定，新端點依照測

量結果加入最近的叢集，直接由叢集領導者管轄。在新節點加入超級端點或叢集領導者的管轄後，會將自己擁有分享資源的引索發佈給超級端點或叢集領導者。同時，超級端點會將收到的資源引索轉錄到叢集領導者上。

對於端點的離開，SSIF 根據不同種類的端點採取不同的動作；一般端點的離開對於拓樸整體的影響最小，只需向超級端點發出離開訊息，超級端點清空該端點相關資訊後，一般端點便可離開。而負責維持拓樸的超級端點和叢集領導者在離開時，必須通知其他的端點。超級端點通知備選端點後才可下線。備選端點會告知同廣播領域中所有的一般節點，更新超級端點的位址。而叢集領導者離線時，叢集領導者的備選端點必須通知同叢集中的所有超級端點，更新叢集領導者的位址。

3.3 SSIF 的合併與遷移

當 SSIF 中的一個叢集過大或過小時，SSIF 必須進行遷移或合併的動作。在遷移的情況下，從原有叢集中被移出的端點將自組成另一個新的叢集。SSIF 會在新叢集和原有叢集之間建立邦聯關係，擁有邦聯關係的叢集在資料整合時將共享資訊。在合併的情況下，端點成員過少的叢集將所有的成員移至其他的叢集。不論合併還是遷移，SSIF 以步進的方式周期性地一個一個移動端點，減少叢集反覆建立和摧毀的震盪現象發生。

3.4 資源搜索

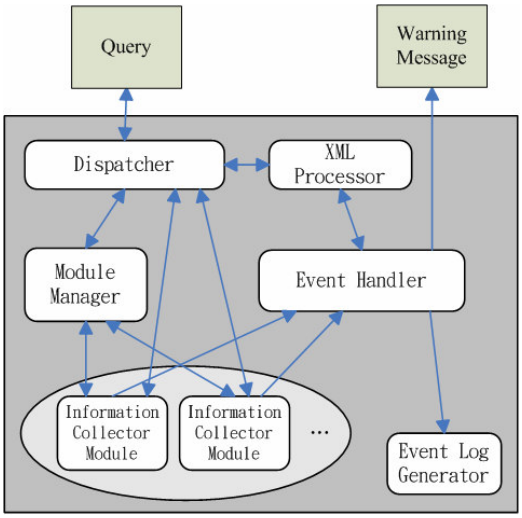
藉由使用 SSIF 的網路結構，監測資料的檔案搜尋以區域性的方式查詢；當某一端點發出搜尋請求時，必先由所屬區域的超級端點負責回覆。如果超級端點無法在區域中搜尋到符合請求的結果，則區域管理者將搜尋請求轉送更高階層的叢集領導者。當叢集領導者也無法搜索到檔案資料時，則叢集領導者將搜索訊息轉送到其他叢集領導者中搜索。此種漸進式的搜尋方法，比傳統的氾濫式(flooding)搜尋更能節省網路頻寬的使用量。

3.5 監測代理器系統架構

監測代理器(此後將簡稱為「代理器」)之架構，如圖二所示，分為六個元件：

A. 分配者元件(Dispatcher Component)：此元件最主要是將接收到的指令，依照種類分配給予不同的元件(ICM 或 Module Manager)執行，並負責將執行結果傳回給使用者。

B. XML 處理器元件(XML Processor Component)：代理器收送的指令和資料，皆是以 XML 文件的方式描述。在代理器架構中，XML 處理器元件主要達成產生 XML 文件、轉換 XML 文件等，與 XML 文件相關的操作。



圖二、監測代理器架構圖。

C. 事件紀錄產生器元件(Event Log Generator Component)：負責紀錄代理器中發生的事件(event)，當代理器進行特定的動作而產生事件時，如載入模組、模組執行錯誤，事件紀錄產生器便將此類訊息紀錄於本地端檔案之中。

D. 事件處理者元件(Event Handler Component)：當代理器內部發生事件時，發生事件的元件便通知事件處理者元件，事件處理者會依照事件的種類，將訊息發送給事件紀錄產生器元件紀錄，並發送警告訊息給特定使用者。事件處理者元件發送的訊息，可能會交由 XML 處理器元件轉換為 XML 文件，或是使用特殊的格式，如 e-mail，傳送給使用者。

E. 模組管理者元件(Module Manager Component)：模組管理者元件是負責管理資料收集模組(Information Collector Module；ICM)的載入和卸下。透過模組管理者元件，可以簡便的管理 ICM。而對於直接存取監測資料的要求，則是由分配者元件直接與 ICM 溝通互動。

F. 資料收集模組(Information Collector Module；ICM)：ICM 在代理器中是扮演收集監測資料的角色，以模組化的方式設計，以因應不同的監測與應用。藉由分配者元件的管理，使用者可以依照情況選擇適當的模組來收集特定的監測資料。ICM 是可以客製的(customized)，只要撰寫者遵守程式撰寫規則，即可寫出自己的 ICM，並交由代理器載入系統執行。如此增加整個代理器的延伸性(scalability)及適用性(flexibility)。

從功能性分類，代理器可以分成網路管理、模組管理、與錯誤處理三部分：

A. 網路管理：主要負責需傳送、或接收的資料和

要求。分配者元件屬於此類。

B. 模組管理：注重如何取得和管理監測資料。模組管理者元件、XML 處理器元件、與資料收集模組歸於此類。

C. 錯誤處理：針對代理器在運行時可能發生的錯誤，進行處理。事件紀錄產生器元件和事件處理者元件屬於此類。

在網路部份中，本代理器使用 XML 文件做為資料交換的媒介。XML 文件的優點是，透過 XML 的標籤語言的描述，資料可以跨系統地交換。使得代理器在異質性的網路系統中，也能順暢的運行無礙。

在監測部分，代理器使用模組化的方式收集監測資料，使用者可動態地管理 ICM 以增加效率。ICM 的運行模式是使用執行緒(thread)定期擷取監測資料，並且將監測資料記錄存於本地資料庫之中。

代理器提供一套 API 給使用者自行撰寫資料收集模組。使用者可以撰寫客製化的 ICM，並交由代理器載入執行監測動作。同時，本研究的實作也提供了遠端部署(Remote Deployment)的功能。使用者可從遠端的節點上，傳輸一個資料收集模組給予本地端的代理器，並交由代理器載入執行。

4. 監測代理器實作

監測代理器的實作包含三部分：監測代理器、遠端管理程式和遠端管理 API。遠端管理程式可以遠端操作監測代理器，並且提供遠端佈署監控模組的功能。使用者可將 ICM 交由遠端管理程式傳送至監測代理器載入執行。遠端管理 API 是供給軟體開發者使用，有效地利用監測代理器提供的監測資料。監測代理器已完成實作，其實作規格如表格一所示。

表格一、監測代理器實作一覽。

程式	監測代理器	遠端管理程式	遠端管理 API
平台	Linux (Kernel 2.2 以上)	Microsoft Windows	Java2
類型	Daemon	圖形介面程式	API
語言實作	C	C++	Java2

對於代理器的實作，本章分成三部分介紹，第一部分介紹代理器對於 ICM 的管理方式，第二部分介紹 ICM 的內部架構，第三部分介紹代理器所使用的 XML 文件格式。

4.1. 監測代理器的管理機制

代理器可供操作的指令分成模組管理與資料及操作管理兩類：

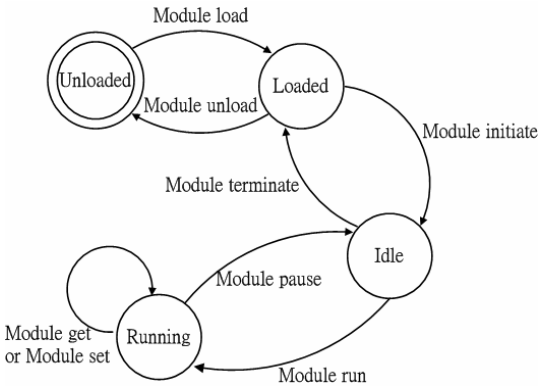
A. 模組管理：管理 ICM 的相關指令，包含 Module load、Module unload、Module status 和 Module query 等指令。

B. 資料及操作管理：收集監測資料的相關指令，也包含設定 ICM 的監測方式。此類指令包含 Module initiate、Module run、Module pause、Module terminate、Module set 和 Module get。

Module load 指令為載入 ICM 至代理器中，ICM 被載入前，代理器會對 ICM 作驗證的動作，確保 ICM 的可執行性及安全性。Module unload 指令為從代理器中卸下 ICM。代理器先確認要卸下的 ICM 是停止執行的，否則拒絕卸下，在卸下 ICM 後，代理器將釋放與此 ICM 相關的資源。

Module status 與 Module query 是查詢 ICM 狀態的指令，Module status 可指出特定 ICM 是否已被載入，Module query 則是列出代理器中所有已被載入的 ICM。

Module initiate 與 Module set 皆是設定 ICM 的參數。參數控制 ICM 的監測行為，如多少時間為一週期擷取監測資料等等。Module initiate 是 ICM 進行初始化(initiating)時執行的指令。Module set 則是在 ICM 運行時改變參數的指令。Module run 是命令 ICM 開始依照參數的設定進行監測動作。使用 Module set 後並不需要再使用 Module run，ICM 會自動依照新的參數值進行下一次的監測動作。



圖三、監測代理器指令操作順序示意圖。

Module pause 的作用為暫時停止執行中的 ICM，其時間單位以秒計算。Module terminate 指令為停止執行中的 ICM。Module get 指令是取得監測資料和 ICM 參數的指令，使用者也可以經由 Module get 取得周期性統計的監測資料，如五分鐘內記憶體的平均使用容量。

所有的指令以 XML 文件的格式傳送給代理器。代理器對於 ICM 的管理和操作是有一定的狀態轉換順序，如圖三所示。以下對圖三中不同的狀態加以說明：

A. 未載入(Unload)：代理器的初始狀態。在這個

階段中，代理器並未載入 ICM。

B. 已載入(Loaded)：當代理器接收到指令，要求載入特定 ICM 時，分配者層便尋找指定的 ICM(在工作目錄中)並載入。在實作中，代理器最多可同時載入 20 個 ICM。

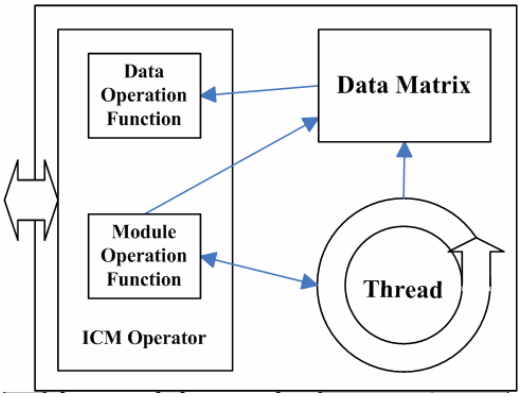
C. 閑置(Idle)：ICM 被載入，且經由 Module initiate 指令設定監測的資源項目、和抓取資料的週期等監測行為之後，便進入此狀態之中。當 ICM 停止運作後，也是回到此閑置狀態。

D. 執行中(Running)：ICM 依照在初始化中設定的項目，啟動執行緒並使用執行緒開始定期收集監測資料。在這個狀態中，代理程式可使用 Module get、Module set、Module pause、以及 Module terminate。當使用 Module terminate 時，ICM 會銷毀執行緒並停止運行，回到閑置的狀態。

圖三中並未列出的 Module status 和 Module query，這兩項指令在任何階段都可使用。表格二列出可供操作的指令。

表格二、監測代理器指令總匯。

	功能名稱	描述
模組管理	Module load	載入 ICM。
	Module unload	卸下 ICM。
	Module status	查詢特定 ICM 是否已載入。
	Module query	列出代理器所有已載入的 ICM。
資料及操作管理	Module initiate	設定 ICM 的參數。
	Module run	啟動 ICM。
	Module pause	暫停 ICM 的監測動作。
	Module terminate	終止 ICM 的監測動作。
	Module set	ICM 在執行時改變其參數。
	Module get	取得監測資料。



圖四、ICM 架構圖。

4.2. ICM 內部架構

ICM 是負責收集資源狀態資料的模組，代理器同時提供數種不同的 ICM 以因應不同的監測需求，表格三顯示預設提供的 ICM。ICM 架構可分為下列三部分，如圖四所示：

A. ICM Operator：負責接受代理器指令，是整個 ICM 的操作介面，同時也管理 ICM 內部元件。上小節所提及的代理器資料及操作管理功能是實作在 ICM Operator 之中。而 ICM Operator 又分成兩部份功能：

   i. Data Operation Function：獲取 ICM 所保存的監測資料，此功能對應的指令為 Module get。

   ii. Module Operation Function：管理 ICM 的監測行為，此功能包含的指令為 Module initiate、Module run、Module terminate、Module set、與 Module pause。

B. Thread：專門抓取資源狀態資料的執行緒，本代理器提供的 ICM 中，有部分 ICM 的內部是有數個 Thread 同時在運行的。

C. Data Matrix：存放由 Thread 獲取的監測資料，ICM Operator 也是從 Data Matrix 取得資料以便回應代理器的查詢要求。當 Thread 無法取得最新的資料時，Data Matrix 仍可提供次新的資料給予 ICM Operator。

表格三、代理器提供的 ICM 一覽表。

ICM	監測項目
Memory ICM	系統記憶體相關資訊。
System ICM	CPU、系統開機時間、整體效能等資訊。
Bandwidth ICM	測量兩節點之間的 available bandwidth。(兩節點都需要執行監測代理器且包括 Bandwidth ICM)
Disk ICM	硬碟分割區 (partitions) 相關資訊。
Network ICM	監測網路效能。
Process ICM	監測特定程序。

4.3. ICM 佈建與發佈

當使用遠端佈署監控模組的功能時，則是將 ICM 以 Base64 編碼法編碼，嵌入在元素 file 的標籤中，加入 XML 文件。當代理器收到含有元素 file 的 XML 文件時，將把元素 file 中的內容逆向編碼，解出 ICM。轉換後的 ICM 放置於代理器的工作目錄中，等待進一步的操作、如 Module load、Module initiate 等。

我們在實作中，使用兩種不同容量(size)的 ICM 進行遠端佈署的功能測試，傳輸的資料包含 ICM 本體和命令代理器執行 ICM 的腳本(script)，如表格四所示，從遠端傳送 ICM 到代理器的時間分別為 2 秒

和 6 秒，當 ICM 傳送完畢時，代理器可依照與 ICM 隨附的腳本執行載入、啟動 ICM 的操作。

表格四、ICM 傳輸時間。

ICM 容量(KB)	傳輸時間
30 KB	2 sec
103 KB	6 sec

4.4. 監測代理器 XML 文件格式

代理器是以 XML 文件當作資料交換的媒介，本節將介紹代理器使用的 XML 文件格式。圖五為向代理器查詢記憶體資源狀態的 XML 文件，圖五(a)為向代理器查詢的文件，圖五(b)則是由代理器處理後傳回的文件。代理器收到的圖五(a)文件解析執行相對應的操作後，便將結果至入圖五(b)文件之中。

```
<?xml version="1.0"?>
<module>
        <icm>memory</icm>
        <func>get</func>
        <result></result>
        <items>
                <item>
                        <type>MEMTO</type>
                        <value></value>
                </item>
                <item>
                        <type>MEMFR</type>
                        <value></value>
                </item>
        </items>
</module>
```

(a)

```
<?xml version="1.0"?>
<module>
        <icm>memory</icm>
        <func>get</func>
        <result>SUCCESSED!!</result>
        <items>
                <item>
                        <type>MEMTO</type>
                        <value>523760</value>
                </item>
                <item>
                        <type>MEMFR</type>
                        <value>124976</value>
                </item>
        </items>
</module>
```

(b)

圖五、代理器所使用的 XML 文件。

所有的指令都需要以 XML 文件的格式傳送給代理器執行。每一種指令的 XML 文件基本格式都是相同的，但是對於需要夾帶大量資料的指令，如 Module initiate、Module set、Module get 等，則是在元素(element)result 之後添加額外的元素，如元素 items 表示要查詢的項目、元素 param 表示要傳入的參數。在目前的實作中，一份 XML 文件只能包含一個指令。

在圖五 (a)的例子中，使用者使用 Module get 指令，向代理器查詢記憶體總容量，與記憶體可用容量。這兩者的分別以 MEMTO 和 MEMFR 的項目名稱填入元素 type 中。而代理器回應時，將資料放入元素 type 的同屬(sibling)元素 value 中傳回。圖五的 XML 文件中，出現的元素及其描述如表格五所示。

如圖六所示，元素 file 內含經過 Base64 編碼的 ICM 程式碼，整個元素 file 是接在元素 result 之後。

```
<?xml version="1.0"?>
<module>
        <icm>memory</icm>
        <func>get</func>
        <result>SUCCESSED!!</result>
        <file>KJIEWHDU0974HBRF8u50u5oythj654oghnfeirjtruohUIJIhuhhgfef</file>
        <items>
                <item>
                        <type>MEMTO</type>
                        <value>523760</value>
                </item>
                <item>
                        <type>MEMFR</type>
                        <value>124976</value>
                </item>
        </items>
</module>
```

圖六、傳送 ICM 的 XML 文件。

表格五、代理器的 XML 文件元素解說。

元素名稱	描述
module	指定要操作的 ICM。
func	指定要進行的指令。
result	指令執行的結果，如失敗則會列出失敗原因。
items	項目的集合
item	單一項目。
type	表示此項目的名稱。
value	表示此項目的值。

5. 結論與未來工作

監測系統藉由使用 SSIF 增加網路拓樸的穩固性，有效地分散資源管理所需的開支，而且能自動調整拓樸結構，維持監測系統的執行效率。

本文提出的監測代理器架構，實作上提供 API 給予使用者開發監測模組和管理監測代理器。同時以 XML 文件作為資料交換媒介，並使用模組化的方式擷取系統的資源狀態資料。監測代理器提供即時性的監測資料，也可提供周期性統計的監測資料。監測代理器使用 XML 文件可簡化資料的處理難度；而模組化架構提供使用者擴充新的監測模組，增加代理器的適用性(flexibility)。

針對未來的工作，本研究繼續擴充監測代理器，有下列的方向：

A. 改進 XML 文件格式：目前代理器所使用的 XML 文件格式，只能一個文件包含一個指令。未來將利用 XML 文件中的命名空間(namespace)特性，使得使用者在一個 XML 文件中可同時對多個 ICM 進行操作。

B. 擴充認證及加密功能：防止來路不明的使用者隨意獲取監測資料。同時，加強 ICM 驗證功能，避免代理器載入損壞，或含有惡意執行碼的 ICM。

C. 建立圖形化管理系統：建立一套網站方式的圖形化管理系統，使用者經過認證登入、以簡單的選取動作即可得知各個節點的即時狀態資

訊；且監測資料是以圖表的方式加以呈現。

D. 跨平台移植：將代理器移植到 Windows 平台上。

6. 參考文獻

[1]. S. Ratnasamy et al., “A Scalable Content-Addressable Network”, in *Proc. of ACM SIGCOMM 01*, pp. 161-172, 2001

[2]. I. Stoica et al., “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, in *Proc. of ACM SIGCOMM 01*, pp. 149-160, 2001

[3]. A. Rowstron and P. Druschel, “Pastry: Scalable Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems”, in *Proc. of IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, pp. 329-350, 2001

[4]. Ben Y. Zhao et al., “Tapestry: A Resilient Global-scale Overlay for Service Deployment”, *IEEE J. Sel. Areas. on Comm.*, Vol. 22, No. 1, Jan. 2004

[5]. Y. Jiang, C.-K. Tham, and C.-C. Ko, “Challenges and approaches in providing QoS monitoring”, *Intl. J. of Network Management*, Vol.10, No. 6, pp. 323 – 334 Nov./Dec. 2000

[6]. A. H. Asgari et al., “Building Quality-of-Service Monitoring Systems for Traffic Engineering and Service Management”, *J. of Network and Systems Management*, Vol. 11, No. 4, pp. 399 - 426, Dec. 2003

[7]. D. W.-K. Hong and C. S. Hong, “A QoS management framework for distributed multimedia systems”, *Intl. J. of Network Management*, Vol. 13, No. 2, March/April 2003

[8]. I. Satoh, “Configurable Network Processing for Mobile Agents on the Internet”, *Cluster Computing: the J. of Networks, Software Tools and Appl.*, Vol. 7, No. 1, pp. 73 - 83, Jan. 2004

[9]. R. V. Renesse, K. P. Birman, and W. Vogels, “Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining”, *ACM Trans. on Computer Systems*, Vol. 21, No. 2, pp. 164 - 206, May 2003

[10]. C. Estan, S. Savage, and G. Varghese, “Automatically inferring patterns of resource consumption in network traffic”, in *Proc. of ACM SIGCOMM 2003*, Vol. 33, No.4, pp. 137-148, Oct. 2003.

[11]. L. Cherkasova et al., “Measuring and characterizing end-to-end Internet service performance”, *ACM Trans. on Internet Technology*, Vol. 3, No. 4, pp. 347 - 391, Nov. 2003

[12]. D. W.-K. Hong and C. S. Hong, “A QoS management framework for distributed multimedia systems”, *Intl. J. of Network Management*, Vol. 13, No. 2, March/April 2003

[13]. D. Harrington, R. Presuhn, and B. Wijnen, ”An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks”, *IETF STD0062*, http://www.rfc-editor.org/rfc/std/std62.txt, Dec. 2002

[14]. T. Klie and F. Strauβ, “Integrating SNMP Agents with XML-Based Management Systems”, *IEEE Comm. Mag.*, pp. 76~83, July 2004

[15]. P. Yalagandula, M. Dahlin, “A Scalable Distributed Information Management System”, *ACM SIGCOMM’ 04*, Aug. 30-Sept. 3, 2004.