利用JAVA並結合XML和RSS以設計Web障礙通告管理系統

楊欣哲
東吳大學資訊管理學系

謝明宏
東吳大學資訊管理學系

摘要

障礙管理(Fault management)，是ISO 7498-4所定義網路管理的五大重要功能之一。其中障礙的偵測(fault detection)及通知(notification)為整個障礙管理極為重要的前置處理程序，它標示著網管人員處置障礙所能運用的時間長度，並且關係著後續障礙處理的進行及品質。

隨著網際網路的快速發展與網路環境的日益複雜，快速的障礙辨識和過濾重覆的障礙訊息，已成為提升障礙管理品質的重要課題。本篇論文將提出一種新的障礙通告發佈系統，稱為「RSS障礙通告發佈系統」(RFNS: RSS Fault Notification System)，藉由系統的「障礙政策制訂模組」以制定障礙通告的相關政策(policy)，作為進行障礙通告的發佈依據。利用「障礙萃取器」來過濾重覆出現的障礙通告，藉以降低因發佈障礙通告所需耗用的網路頻寬及系統資源；透過「RSS處理器」將障礙通告轉換成RSS格式，並以新聞的方式快速而即時的發佈給相關網管人員。本文將實作RFNS系統，並藉由實驗模擬進行績效分析。最後，實驗結果證實經由RFNS的執行，確實能夠達到大幅降低因障礙通告發佈所佔用的網路頻寬與減少其檔案大小，同時亦經由縮短障礙偵測時間，進而降低整體障礙處理所花費的時間。此外，並精萃了障礙通告的內容，讓網管人員能即時地和正確地獲得相關的障礙資訊。

關鍵字：XML、網路管理系統、RSS、網路管理資訊庫、障礙管理

Using Java and Combinations of XML and RSS to Design Web-based Fault Notification Management

Shin-Jer Yang
Department of Computer Science and Information Management,
Soochow University

Ming-Hong Hsieh
Department of Computer Science and Information Management,
Soochow University

Abstract

The fault management is one of five essential network management tasks defined by ISO 7498-4. Fault detection and notification are the key pre-process of all the fault management, it is noted that the network manager can use the disposal time of the faults, and is also related to the activity and quality of follow-up processing.

With the fast development and the increasingly complex environment of Internet, rapid fault identification and duplicate messages filtering is one of essential subject to enhance the fault management quality. This paper proposes a novel fault notification issue system, called RFNS. The "Fault Policy Refine Module" formulates the related policy of fault notification via RFNS system. Also, the RFNS system is to solve the fault notification issue and to utilize "Fault Extractor" module for filtering duplicate fault notification in reducing network bandwidth and system resources of publishing fault notification. Then, the notification will convert into the RSS format through the "RSS Processor" and immediately issue the related network manager by news. Also, we will implement the RFNS system and perform experimental simulations for performance analysis. Consequently, the final results indicate that lower the wasted network bandwidth, lessen the file size of fault notification and shorten the total amount time in faults detection and processing through the implementation of RFNS. In addition, the RFNS system extracts the main content of fault notifications, and therefore the network manager can obtain real-time and accurate fault information.

Key words: XML, Network Management System, RSS, Management Information Base, Fault Management

壹、緒論

一、研究背景與動機

傳統的網路管理系統(Network Management System, NMS)多以SNMP(Simple Network Management Protocol)做為主要的通訊協定，其最大的優勢在於容易實作且絕大多數的網路設備皆支援（Ju et al. 2002），遂擁有在異質產品中，有著良好溝通的相對優點。但自80年代的WWW(World Wide Web)問世後，網際網路(Internet)即因擁有標準的通訊協定以及開放、多元的平台，而呈現爆炸性般的快速成長；再加上90年代起寬頻多媒體時代的來臨，網路世界更是一日千里，蓬勃發展。為了因應複雜的網路環境，各式各樣的網路設備(Network Equipments)組合便應運而生，而SNMP協定過於簡單與容易，卻也成為了最大缺點，導致其在擴充性(Scalability)與效率(Efficiency)上更顯捉襟見肘。

基於上述SNMP的缺點，許多文獻便提出了整合SNMP與XML(eXtensible Markup Language)的網管系統，藉以提昇網路管理的效率與增進其擴充的彈性（Klie and Straus 2004; Yoon et al. 2003）。經由上述文獻的探討，一再的驗證經由SNMP與XML的整合，可以做為現有以SNMP做為網管基礎協定所產生缺失的改善方法。

障礙管理(Fault Management)，是ISO 7498.4所定義網路管理的五大項目中，非常重要的功能項目之一，其中障礙的偵測(Fault Detection)及通知(Notification)更是整個障礙管理極為重要的前置環節，它標示著網管人員處置障礙所能運用的時間長度，並且關係著後續障礙處理的進行及品質。而隨著網路環境的日益複雜，網管人員必須隨時處理由網路設備(Network Elements)所發送的大量封包訊息，而如何在這些訊息中有效而快速的執行障礙封包的辨識以及過濾重覆的障礙訊息，已成為提升障礙管理品質的重要課題。

二、研究目的

而網路管理系統的主要目標，在於確保全體的網路設備都能夠提供一定水準的服務，而為了達成此一目標，網管人員就必需隨時監看、控管已連結之設備皆能正常運作（Sirajuddin and Sqalli 2006）。XML-based的網管系統(XML-based Network Management; XNM)，主要是由下列幾個元素所構成，分別為：SNMP/XML gateway、SNMP agents、以及XML格式的網管資料輸出，其中在SNMP/XML gateway的領域裏，更有眾多的學者專家鑽研其中，並提出了許多相關的研究報告（Bourges-Waldegg and Hoertnagl 2005; Wu and Yan 2008; Yoon et al. 2003）。但在網路環境日益複雜的今日，網路障礙發生的機率，經常和網路架構的複雜度成相對比率的成長，且障礙排解的速度與正確性，又往往成為左右整體網路效能與品質的重要關鍵因素。因此，要如何有效率的從XNM所輸出的大量XML資料中，快速的辨識出那些是屬於障礙訊息、障礙訊息的種類，以及在最短的時間內告知相關的網管人員進行後續的處理動作，已成為一個十分重要的課題。有鑑於此，本論文主要的研究目的有下列幾項：

1. 以XML-based的網路管理系統為基礎，並針對所輸出之大量網管相關之XML資料中，設計一套能夠快速辨識那些屬於障礙訊息、萃取有效障礙訊息，以及在最短的時間內告知相關網管人員進行後續處理動作的障礙通告管理系統，稱為「RSS障礙通告發佈系統」(RSS Fault Notification System, RFNS)；
2. 建置障礙政策制訂模組(Fault Policy Refine Module)，經由定義需發布障礙通告的相關參數，如產生訊息之網路設備、物件識別碼(Object Identifier, OID)、物件可容許範圍…等，藉以辨識出需發布障礙通告的有效障礙訊息；
3. 建置障礙萃取器(Fault Extractor)，過濾重覆發布之有效障礙訊息，以降低因發佈障礙通告所需耗用的網路頻寬及系統資源，精萃障礙通告之內容；
4. 建置RSS處理器(RSS Processor)，將障礙通告轉換成RSS格式，並以新聞的方式快速而即時的發佈給相關網管人員；
5. 將上述所提之RSS障礙通告發佈系統架構，使用JAVA語言實作開發，並藉由模擬實驗進行績效分析，以驗證本文所提出的障礙通告管理系統架構確實能夠達到大幅降低因障礙通告發佈所佔用的網路頻寬與減少其檔案大小。此外，並精萃了障礙通告的內容，讓網管人員能即時的和正確的獲得相關的障礙資訊。

貳、文獻探討及相關研究

一、SNMP的障礙管理

SNMP協定提供Trap指令，讓網路設備在發生問題時，透過UDP的162埠主動的發出警報，通知網管系統該設備發生了異常狀態或某些重要事件，在RFC 1157中定義了七種標準的Trap警示類型，其中enterpriseSpecific類型是為了提供網管人員根據其自身需要所自行定義的專屬警示類型。我們都知道於OSI的七層架構中，其傳輸的封包是由各層次分別將其資料加入整體的封包內容中，而每層封包所定義的欄位格式，就稱之為PDU(Protocol Data Unit)。當PDU記錄中的標準Trap類型定義為enterpriseSpecific時，則此時便需要參考在PDU記錄中的特殊Trap類型所定義的內容，而網管系統也必需為該設備建立專用的MIB，以了解在特殊Trap類型中所記錄數值的含義。綜觀上述說明，發現Trap有下列問題：

1. 因為SNMP採用UDP協定，很有可能Trap封包在傳送過程中丟失，所以無法保證網管系統會確實收到相關訊息；
2. 當網路設備發生重大故障時，例如系統當機、毀損…等，導致Trap封包根本就無法由網路設備自行發送，而使網管系統無從得知有重大事件發生；
3. RFC 1157所定義的標準Trap類型過於簡單，雖然提供了可讓網管人員自行定義警示類型的enterpriseSpecific，但對於MIB中物件型態屬於數值屬性之跨越門檻值(Threshold)的事件產生，就無法完全以Trap的PDU格式來加以定義。

所以障礙管理無法只依靠Trap就能達成，要全面的對網路進行障礙偵測，網管系統

就必須對監控的設備定期輪詢(polling)，輪詢是指網管系統對設備提出讀取(get)需求，而設備則會將所有的MIB資訊透過回應(GetResponse)指令，傳送給提出需求的網管系統，但MIB所描述的資訊眾多，網管人員需耗費資源與時間對相關資訊進行解讀，這也是使用輪詢所必需面臨的重大問題，圖1為SNMP障礙偵測架構說明圖。

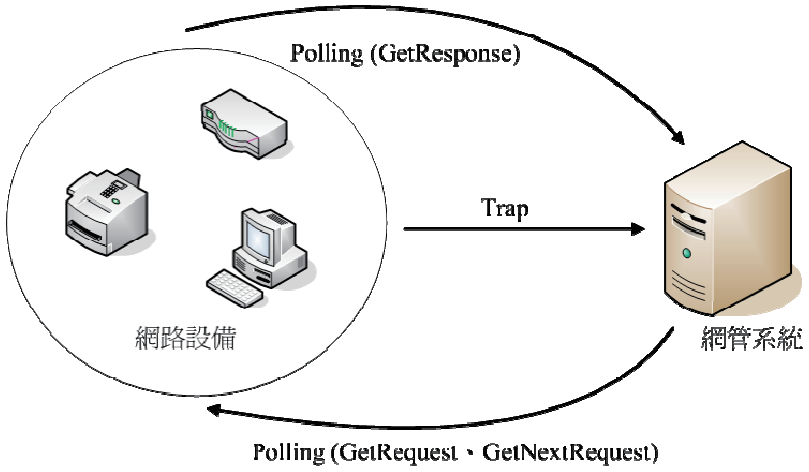

圖1：SNMP 障礙偵測架構圖

障礙管理主要包含了障礙的偵測、隔離(Isolation)、通告(Notification)、障礙排除修復(Correct)，以及障礙記錄(Log)的儲存與維護(Maintain)（Jiang et al. 2005; ISO 7498-4）。其中障礙的偵測與通告是障礙管理中非常重要的前置作業，因為網路障礙一旦發生，若無法即時加以適當的處置，則可能就會如野火燎原般，快速漫延，甚至癱瘓整個網路系統。要達成障礙管理的目標，就必需藉由不斷收集來自網路上各控管設備之狀態資訊，因此障礙管理系統所要處理的將是數量龐大的封包訊息，而且必須能夠即時的進行判別與分析，並提出解決的方法，這也是障礙管理所面對的最大挑戰。因此如何掌握關鍵的黃金救援期，縮短障礙管理的偵測與通知時間，便成為相當值得研究的重要課題。

Russell Arrowsimth 等人在2000年提出建立網路管理的警示通告政策資料庫(Policy-based Alarm Notification Database)，將網路管理系統所產生的警示訊息，經由應用程式與資料庫中的相關網管政策進行過濾比對，找出其處置的對應策略，藉此加速網路障礙的排除及修復，以提升網管的整體品質與效率。本論文將引用此構想，建置一套障礙通告控管資料庫(Fault Notification Control Database, FNCD)，依據網路設備MIB(Management Information Base)所定義的物件，將需要即時通告的物件，記錄在FNCD中，並設定該物件之相關屬性，包括物件的正常範圍說明(如：通訊埠封包流量的最大上限)，發生異常狀態後之標準處置說明等，藉由該資料庫的比對與過濾，利用RSS的技術即時的將障礙訊息通告給相關網管人員進行處理，達到縮短障礙處理時間的目標。

二、XML-based網路管理系統

在第一章中我們曾經提到，SNMP-based的網管系統有著擴充性(Scalability)差與效率(Efficiency)不佳的問題，於是借助XML技術所提出的XML-based網管系統遂應運而生。XML-based網路管理系統利用XML技術來執行網管任務，圖2說明XML-based網路管理系統中，最重要的SNMP/XML gateway的架構圖（Yoon et al. 2003）。

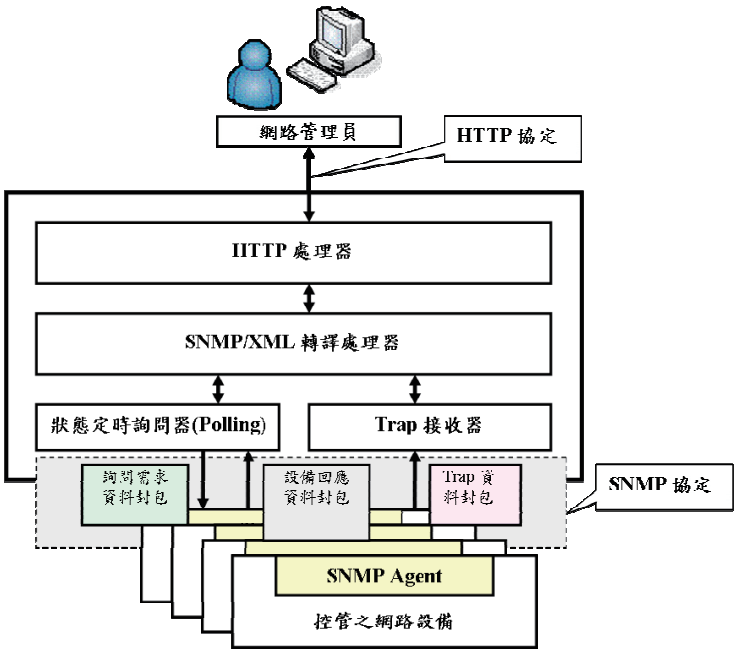

圖2：SNMP/XML gateway的架構圖

在XML-based 網管系統中，SNMP/XML gateway依循系統所訂定之輪詢(Polling)排程，使用SNMP協定與所控管之網路設備的SNMP Agent進行溝通，或是接收來自所監控之網路設備所傳來之Trap資料封包，gateway再經由本身的SNMP/XML 轉譯處理器，依據已定義好的DTD或XML Schema，將接收的SNMP封包內容轉換為標準的XML文件，再經由HTTP協定將XML文件傳送至使用者的管理介面。以POSTECH團隊所提出之XML-based網管系統為例（Yoon et al. 2003），它提供了兩種方式將網管資料由SNMP/XML gateway傳送給網路管理者，第一種方式是被動的由網管人員提出查詢需求，網管人員依照圖3的URI格式，經由HTTP協定向SNMP/XML gateway提出需求，gateway便會依據需求傳回XML格式的MIB資訊；第二種則是網路設備因發生重大事件而產生Trap封包時，當SNMP/XML gateway接收到相關訊息後，會主動的將該MIB資訊轉換為XML格式的文字檔，立即傳送給網管人員，圖4為Trap觸發後之障礙通告相關流程說明。本論文將提出延伸XML-based網管障礙通告系統，藉由其擴充性與較佳的效率特性，並搭配XML格式的RSS文件，以縮短障礙處理的時間。進一步提昇障礙管理品質。

SNMP action	HTTP action	Note
Get	http://(gateway_address)/SNMP?host = *agent_name* &community = *community_name*&operation = *get* &path = *node_name*	HTTP GET

圖3：XML-based網管系統的查詢需求URI格式

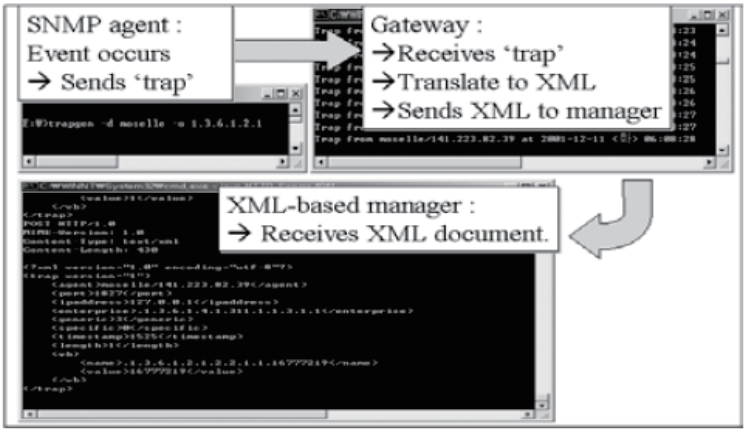

圖4：Trap觸發後之障礙通告相關流程

XML是由Wide Web Consortium(W3C) 在1998年為了文件交換所制定的規範，XML是一種標記語言(Markup Language)，使用者可以利用Document Type Definition(DTD)或是XML Schema來自行定義屬於自己的文件結構。XML本身可以藉由HTTP的通訊協定將大量的資料傳輸至網路的另一節點，且有許多標準的API(Application Program Interface)及免費的軟體可提供給開發者非常容易的就能進行各種用途之應用實作。Martin-Flatin, J. P. (Martin-Flatin, J. P. 2000)在他的研究中，首先提出了SNMP的MIB轉換為XML格式的模型，並且描述了以XML/HTTP based的通訊方式來進行SNMP的MIB與XML之間轉譯的優勢。他並提出了兩種轉譯的模型，分別為model-level mapping與metamodel-level mapping。在model-level mapping中，每一個SNMP的MIB皆擁有屬於自己特定之DTD；而在metamodel-level mapping中，則是定義一個可適用於所有的SNMP MIBs的通用雛型(Generic Prototype)。2003年的五月，IETF在Network Configuration (Netconf)下組織了一工作群組(Working Group)，該群組主要在於探討如何將XML技術應用於以IP架構為基礎之網路設備上，並且定義了在組態管理(Configuration Management)上之需求(Requirements)以及提供了關於IETF在使用XML標準下之指導方針。由於現今大部份的網路設備仍舊配置SNMP-based的代理器(Agents)，所以建置SNMP/XML的閘道器，用來做為SNMP MIB與XML格式之間的轉譯器(Translator)，即為現今做為兩種不同資料型態共存的最佳解決之道。

三、RSS架構與相關應用

RSS(Really Simple Syndication)是XML技術的具體應用，它是在Web-based環境下使用Push技術，將即時資訊主動發佈給相關人員〈訂閱者〉，同時透過聚合工具(Aggregator)，將網路上分散於各地的資訊予以整合(Syndication)。RSS已成為現今網際網路最受歡迎的服務之一，網路使用者只需簡單地透過RSS Reader或是支援RSS技術的瀏覽器，就能將所有資料予以整合於單一介面閱覽。圖5為GreateNews 的RSS Reader的使用者介面。RSS之所以成功的原因，可歸功於其通訊方式完全符合標準的HTTP協定，因此比較不會受一般防火牆的封鎖，再加上Web-based的Push技術，Client端的RSS Reader會週期性的詢問Web Server其所訂閱的新聞是否有更新內容，詢問的方式是先從Web Server端接收相關的索引文件(Index Document)。這種型式的文件是遵循XML規範所撰寫，主要的目的是提供給Client端的RSS Reader可以很容易的判別所接收到的索引文件是否有執行過更新動作，藉此以通知訂閱者隨時接收最新的資訊內容。就一個使用者的觀點而言，經由標準的HTTP協定，借助RSS技術，就能在單一介面中，整合Internet上所有感興趣的資訊文章，並且即時的接收到相關內容，無怪乎RSS的出現，帶動了整個Web 2.0的蓬勃發展。

RSS的應用並不只有侷限在個人部落格及新聞資訊的訂閱，有許多的研究論文就將RSS的功能延伸到各個領域，Daniela Bourges-Waldegg 及Christian Hoertnagl在2005年時提出將RSS的技術應用在工作流程控管(Workflow)，並以保險業的理賠流程為範例，說明利用RSS技術可以減少工作流程中因人為因素所造成的延遲與不確定性，並藉此縮短理賠案件的處理週期(Cycle Time)；Jianlin Wu以及Guocong Yan在2008年提出了將RSS技術應用在企業內容管理(Enterprise Content Management, ECM)，並結合Folksonomy，藉此提昇ECM在內容分享(Content Sharing)的效率。

總之，本文提出以XML-based為基礎的網路管理架構，結合RSS的相關技術，藉以提昇網路異常狀況警示通知的速度，並藉此爭取網管人員在處理緊急事故時的黃金時間。

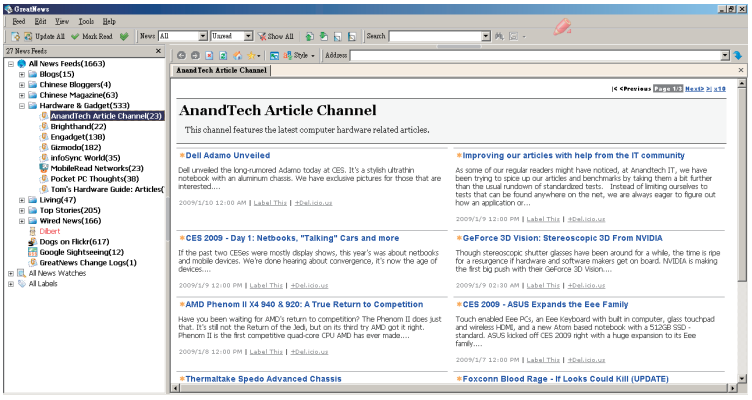

圖5：Trap觸發後之障礙通告相關流程

參、RFNS架構及設計原理

一、RFNS整體架構說明

“障礙管理”經常會產生大量的Trap和Alarm Messages的封包，造成網管人員在進行障礙診斷與修復的行動上極大的干擾。產生大量障礙相關封包的原因，主要在於NMS會透過Polling與設定Trap條件的方式來自動偵測網路錯誤的發生，而在利用Polling方式偵測時，NMS會設定發出Polling的固定週期，而這個週期通常都不會太久，可能五分鐘就發出一次Polling的需求。假設當某一設備發生障礙時，該網路設備的Trap經觸動(Trigger)後，可能就會先發出Trap的封包；NMS在固定週期的Polling偵測中，又會將同樣的障礙資訊傳回給NMS；倘若該障礙的修復動作必須進行一段時間，則在這段期間的Polling動作依舊會持續的傳回該障礙封包。更有可能的狀況是由於該網路設備的障礙產生，導致其它的網路設備也發生了連鎖反應而故障，進而讓障礙的相關封包，呈現倍數的成長，但追根究底卻都可能來自於同一障礙事件的發生。

本論文提出一種新的障礙通告發佈架構，稱為RSS障礙通告發佈系統(RFNS)，在RFNS系統中主要包含了三個模組，分別是障礙政策制訂模組、障礙萃取器以及RSS 處理器，如圖6所示，從下一節開始，我們將詳細說明這三個模組的流程與設計原理。

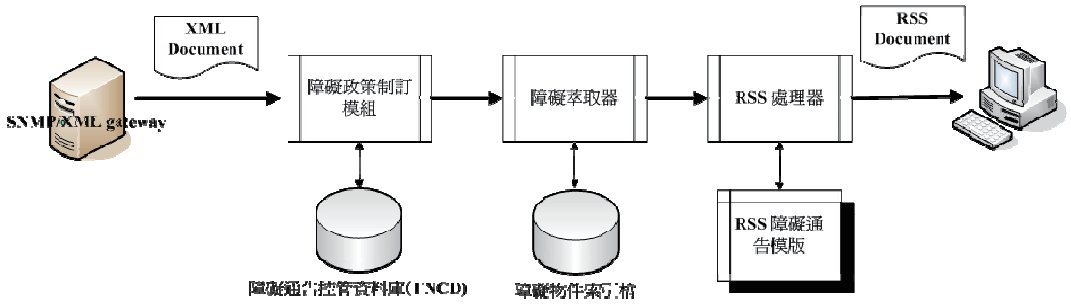

圖6：RFNS架構說明圖

二、障礙政策制訂模組

建構障礙政策制訂模組的主要目的，在於制定網路障礙通告發佈的政策(Policy)，我們知道，當NMS發出Polling的需求後，隨之而來的便是大量的網路設備各相關物件的狀態訊息封包，為了縮短網路障礙發生至NMS發佈通知網管人員的系統執行時間，本論文提出了建構障礙通告控管資料庫(FNCD)，此資料庫建置的主要目的在於設定各網路設備的障礙通告發佈政策，在每一條政策中將會明定障礙通告發佈的條件(網路流量過大、網路連結中斷..等)以及當發生障礙後，該如何處置的相關訊息等內容，表1為FNCD的主要資料欄位說明。當障礙訊息符合FNCD中的記錄條件時，系統就會將預備通知網管人員的相關欄位內容取出(如：故障的網路設備、障礙發生的地點…等)，接著將相關資訊傳入障礙萃取器進行處理。

管理資訊庫(Management Information Base, MIB)是一個樹狀結構的資料庫，主要是用來記錄各個網路節點，如：路由器、工作站、交換器…等設備之屬性與目前狀態，而儲存的內容則是以物件(Object)作為標的物，網管系統經由讀取MIB中所記載之各物件的值，藉以監控整體或各別設備之網路狀態；也可透過修改某些MIB物件之值，進而達到控管網路資源的目的。MIB的樹狀結構，主要是使用管理資訊結構(Structure of Management Information, SMI)及抽象化語法標記語言(Abstract Syntax Notation One, ASN.1)作為定義與描述資料物件的工具（Sirajuddin and Sqalli 2006）。

表1：FNCD欄位說明

欄位名稱	欄位格式	欄位說明	範例
NETWORK_FEVICE	varchar (50)	網路設備的 IP Address	192.168.1.254
OID	varchar (100)	網管物件識別碼	1.6.6.1.2.2.1.2.2.1.10
OBJ_NAME	varchar (100)	網管物件名稱	iflnOctets
OBJ_TYPE	varchar (50)	說明網管物件的類型	INTEGER
MAX_VALUE	varchar (20)	最高上限承受值（數值型態）	100000KB
MIN_VALUE	varchar (20)	最低下限承受值（數值型態）	10KB
OBJ_VALUE	varchar (20)	物件正常狀態值（字串型態）	up
MAX_ERROR_SOP	TEXT	超過上限時之處理流程	重新啟動設備
MIN_ERROR_SOP	TEXT	低於下限時之處理流程	重新啟動設備
OBJ_ERROR_SOP	TEXT	字串型態物件非常之處理流程說明	重新啟動設備

MIB的多樣資料類型是我們在判斷網路物件是否符合障礙通告條件的一項極大變數。在FNCD的設計中，我們將MIB的資料型態區分為兩大類，分別為「數值類別」與「字串類別」，並且依據各資料型態之屬性分別歸納於此兩大類中，表2說明各別資料型態與FNCD所定義的兩大類別之間的關聯性。其中我們將整數型態(Integer)、時間單位型態(Time Ticks)、計數器型態(Counter)、測量器型態(Gauge)…等類型歸類為數值類型；另外將字串型態(Octet String)、空值型態(Null)、網路位址型態(Network Address)、IP位址型態(IP Address)、未知型態(Opaque)等類型歸類為字串類型。

表2：物件資料型態與FNCD兩大類別關聯表

物件資料型態	數值類別	字串類別	具有遞增（遞減）特性
整數型態 (Integer)	√		
字串型態 (Octet String)		√	
空執型態 (Null)		√	
時間單位型態(Time Ticks)	√		√
計數器型態(Counter)	√		√
測量器型態(Gauge)	√		√
網路位址型態(Network Address)		√	
IP位址型態(IP Address)		√	

為了因應多樣的資料型態類別，我們在FNCD的欄位中設計了一個叫做“OBJ_TYPE”的欄位，用來儲存各物件所定義之資料型態，當OBJ_TYPE所儲存的內容為字串型態或網路位址型態等字串類別的資料型態時，則系統將會依據OBJ_VALUE所儲存的值來判斷，是否需要發佈障礙通告。當OBJ_TYPE所儲存的內容為整數型態或計數器型態等數值類別的資料型態時，系統就會參考FNCD關於MAX_VALUE以及MIN_VALUE這兩個欄位值，用來設定此物件的上、下限，並依此來判斷是否需要發佈障礙通告。在此我們必需特別提到幾個具有遞增(遞減)特性的資料型態，分別是時間單位型態(Time Ticks)、計數器型態(Counter)以及測量器型態(Gauge)，這幾個型態其所儲存的值，會不斷因為狀態的改變而累加(或減少)，例如：ipInReceives物件其資料型態便是計數器型態，它代表的是此網路設備其所有的輸入埠(port)累積至目前為止所接收到的封包總數，而非固定的週期片段所接收之封包數；ifOutQlen物件其資料型態為測量器型態，它代表的是此網路設備目前在輸出佇列(Queue)的封包個數，該值可能會因為輸出頻寬品質的良莠不齊，而造成封包數呈現遞增或遞減的現象。因此RFNS在處理這些資料型態時，每次都會將該物件之目前狀態值(如封包數)先予以儲存，並取出前一次的狀態值比對計算，再將計算後之結果與MAX_VALUE以及MIN_VALUE這兩個欄位值進行比較，判別是否有超出該物件在FNCD中所設定之範圍，以此判斷是否需要發佈障礙通告，其相關演算法如圖7所示。

```
1  public void COMPARE_OBJ_VALUE
2  {
3      Declare FNCD as Table
4      Input:
5          Byval strOID as string;
6          Byval strCURRVAL as string;
7      Output:
8          FNCD.OID;
9          FNCD.OBJ_NAME;
10         ERR_SOP;
11
12     Method:
13     BEGIN
14     {
15         SELECT  all fields  FROM  FNCD  WHERE  OID=strOID
16         if  (OBJ_TYPE is numeric type)  {
17             if  IsNULL(MAX_VALUE, MIN_VALUE)
18                 return;
19         }
20         switch(OBJ_TYPE)
21         {
22             CASE  "Counter":    CASE "TimeTicks":   CASE "Gauge":
23                 CurrStatus = strCURRVAL - preStatus;
24                 Update CurrStatus  into  FNCD;
25                 break;
26
27             CASE  default:
28                 CurrStatus = strCURRVAL;
29         }
30         if  (IsNULL(MAX_VALUE)) {
31             if  (CurrStatus < MIN_VALUE)
32                 SHOW_MIN_ERROR_SOP Message();
33         }
34         if  (IsNULL(MIN_VALUE))  {
35             if  (CurrStatus > MAX_VALUE)
36                 SHOW_MAX_ERROR_SOP Message();
37         }
38         if  (OBJ_TYPE is not numeric type)  {
39             if  (strCURRVAL <> OBJ_VALUE)
40                 SHOW_OBJ_ERROR_SOP Message();
41         }
42     }
43 }
```

圖7：FNCD障礙政策比較演算法

另外，針對FNCD資料表的記錄內容，RFNS也設計了一套WEB的使用者介面(User Interface)，藉以進行相關的資料維護作業，該介面使用JavaServer Pages(JSP)作為動態網頁開發工具，後端利用Java Database Connectivity(JDBC)與FNCD資料庫進行連結，圖8為FNCD的資料維護介面。

圖8：FNCD資料維護作業使用者操作介面

障礙政策制訂模組為RFNS系統接收來自SNMP/XML gateway所產生之XML文件之第一道關卡，在第二章的XML-based網路管理系統中曾經說明網路管理者如何取得SNMP/XML gateway所產生的XML格式之MIB資訊，包括了被動的由網管人員提出相關的查詢需求，以及網路設備因發生符合Trap條件的重大事件而主動的發出相關MIB資訊，因此RFNS由SNMP/XML gateway取得XML文件的方式也必須參考上述方法設計。在這裏我們提出了兩種方式以取得SNMP/XML gateway所產生之XML文件：

1. 依據本文第二章圖3中所定義之URI，針對所有控管的網路設備依據該URI格式製作專屬於各設備的HTTP Get指令，再利用批次(Batch)的方式由RFNS自動將設備的XML格式之MIB資訊取出，類似SNMP的輪詢動作。
2. 在XML-based網管系統中，已提供了Trap的自動通告功能，因此當RFNS接收到Trap的XML文件檔時，就會直接將該文件進行XML的解析動作，完成後再將結果直接送到RFNS的第二個模組「障礙萃取器」進行處理，而不需再與FNCD資料庫進行比對，進而縮短障礙處理時間，圖9說明RFNS取得SNMP/XML gateway 所產生之XML文件流程圖，其中實線所標示的流程為使用HTTP Get方法，虛線則為接收Trap自動通告後之處理流程。

當RFNS取得來自SNMP/XML gateway所產生之XML文件後，接著就必須對該XML文件進行解析的動作，因此對於XML的解析功能，亦需建置在處理器中。

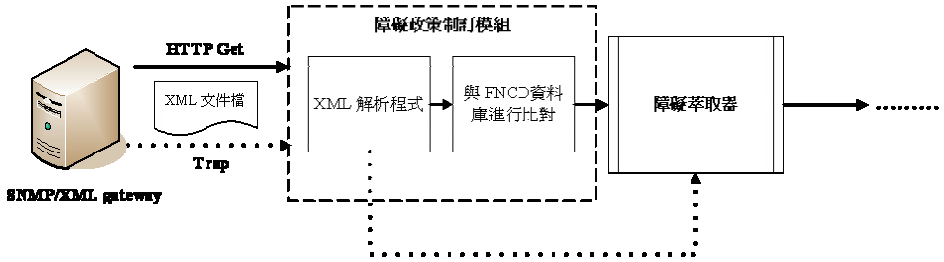

圖9：RFNS取得SNMP/XML gateway 所產生之XML文件流程圖

XML提供了兩種方法用來定義XML文件的結構(Document Structure)，分別為DTDs以及XML Schema，而其中的XML Schema能夠支援較多的資料型態，因此在本系統中將會利用XML Schema來定義XML 的文件結構。另外在SNMP的MIB II所定義的物件種類，幾乎能夠適用於大部份的網路設備，圖10為利用XML Schema定義了在MIB II中的system group，以及在system group裏的“sysDesc”的相關屬性說明（Choi et al. 2003）。經由SNMP/XML gateway所產生的XML 文件範例，其內容如圖11所示（Yoon et al. 2003），該範例主要在於描述MIB II的系統群組(System Group)，其物件識別碼由1.3.6.1.2.1開始，為Microsoft Windows 2000 Version 5.0作業系統的工作站，該工作站的電腦名稱為MOSELLE，本身配備有兩個網路介面卡，其中第一塊網卡單一封包的最大傳輸量為32768(bps)、傳輸速度為10000000(bps)、使用狀態正常，已經運轉了29400398個百分之一秒單位，並且提供了76個服務(services)等相關資訊。

```
<xsd:element name="system">
<xsd:complexType>
  <xsd:all>
    <xsd:element ref="sysDescr" minOccurs="0"/>
    <xsd:element ref="sysObjectID" minOccurs="0"/>
    <xsd:element ref="sysUpTime" minOccurs="0"/>
    <xsd:element ref="sysContact" minOccurs="0"/>
    <xsd:element ref="sysName" minOccurs="0"/>
    <xsd:element ref="sysLocation" minOccurs="0"/>
    <xsd:element ref="sysServices" minOccurs="0"/>
  </xsd:all>
 </xsd:complexType>
</xsd:element>
<xsd:element name="sysDescr">
 <xsd:complexType>
  <xsd:simpleContent>
   <xsd:restriction base="DisplayString_0_255">
    <xsd:attribute name="access" type="xsd:string"
          use="fixed"  value="read-only"/>
   </xsd:restriction>
  </xsd:simpleContent>
 </xsd:complexType>
</xsd:element>
...
```

圖10：XML定義MIB物件

圖11：SNMP/XML產生的XML文件範例

障礙政策制訂模組接收上述由XML格式表示之MIB II資訊檔案後，接下來便要開始進行XML解析工作了。在JAVA開發工具中，提供了JAXP(Java API for XML Parsing) API來支援XML應用程式的開發，其中javax.xml.parsers是主要套件，它提供了DocumentBuilderFactory類別來建立DOM樹狀結構的Document物件，接著使用org.w3c.dom套件所提供之介面(Interface)來處理XML Document物件之元件以及屬性，並且瀏覽以及讀取整個XML文件內容。由於XML文件屬於文字檔格式，所以需要使用JAVA開發工具的java.io套件內的File介面，將檔案匯入到系統中，以利後續之XML解析動作。

三、障礙萃取器

在前面的章節曾經說明產生大量重覆障礙資訊封包的原因，為了避免上述情況的發生，本文提出了障礙萃取器的設計，其完整的運作流程如圖12所示，當該模組接收到障礙封包後，首先會將發生障礙的網路設備之IP address與該設備中發生異常狀態之物件的OID值傳入“障礙物件索引檔”中進行比對，倘若該物件已存在於索引檔中，代表該物件之前已發出過障礙通知，RFNS可忽略該訊息，不予回應；倘若在索引檔中查詢不到該物件，即代表該物件是第一次發出障礙封包，此時RFNS就必需進行障礙通告處理。另外還有一種情況也需要進行通告處理，亦即原存在於“障礙物件索引檔”中的障礙物件，於隨後之Polling狀態偵測時恢復正常，此時就必需發佈“復原通告”，以告知網管人員該設備已經恢復正常。我們在“障礙物件索引檔”中設計一個RESET_FLAG欄位，當系統執行障礙萃取器時，首先會將每筆記錄的RESET_FLAG設定為“R”，當收到障礙的封包已存在於索引檔中，代表該障礙狀態仍舊存在，此時系統會將該筆記錄的RESET_

FLAG設定為“S”，若收到的障礙封包不存在於索引檔中，則系統會將該障礙訊息新增於索引檔中，並將RESET_FLAG設定為“N”，完成了所有的障礙封包過濾比對後，接著障礙萃取器模組會將RESET_FLAG 標註為“N”的所有記錄進行障礙通告處理，隨後該模組也會針對RESET_FLAG 標註為“R”的所有記錄進行復原通告處理，並在處理完畢後，會將標註為“R”的所有記錄由“障礙物件索引檔”中予以刪除。

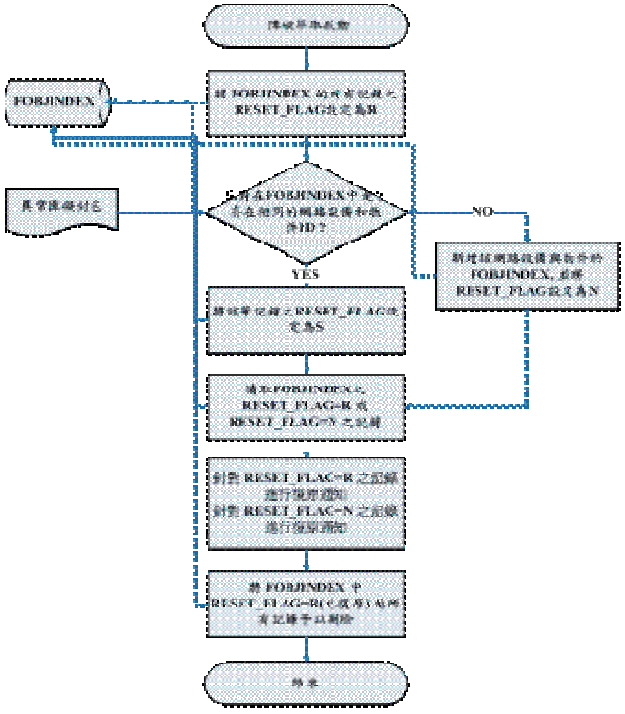

圖12：障礙萃取器運作流程圖

建構障礙萃取器此一模組，主要的目的是過濾掉重覆的障礙通告，避免系統不斷重覆的進行相同的障礙訊息發佈，進而節省系統處理的時間與資源的耗損。經由該模組的運作，藉以降低Trap和Alarm封包的數量，進而提高障礙排除的執行效率，圖13為障礙萃取器模組相關演算法之虛擬碼說明。

```
public void FAULT_EXTRACTOR
{
    Declare FOBJINDEX as Table
    Input:
        Byval  FNCD_ NETWORK_DEVICE as string;
        Byval  FNCD_OID as string;
        Byval  FNCD_OBJ_NAME as string;
        Byval  ERR_SOP as string;
    Output:
        FNCD_OID;
        FNCD_OBJ_NAME;
        ERR_SOP;

    Method:
    BEGIN
    {
        UPDATE FOBJINDEX SET RESET_FLAG='R';

        SELECT   all fileds
        FROM     FOBJINDEX
        WHERE    NETWORK_DEVICE = FNCD_ NETWORK_DEVICE
        AND      OID = FNCD_OID ;

        if (FNCD_OBJECT exist in FOBJINDEX)  {
            UPDATE FOBJINDEX SET RESET_FLAG='S';
            return;
        }
        if (FNCD_OBJECT not exist in FOBJINDEX) {
            Insert  FNCD_OID，NETWORK_DEVICE  into  FOBJINDEX;
            UPDATE FOBJINDEX SET RESET_FLAG='N';
            RSS_Processor_Module(FNCD_OID，FNCD_OBJ_NAME，ERR_SOP);
        }

        SELECT all fileds  FROM  FOBJINDEX  WHERE (RESET_FLAG='N' OR RESET_FLAG='R');

        while (FOBJINDEX.eof)  {
            RSS_Processor_Module(FNCD_OID，FNCD_OBJ_NAME，ERR_SOP);
            FOBJINDEX.movenext;
        }

        DELETE FOBJINDEX WHERE RESET_FLAG='R';
    }
}
```

圖13：障礙萃取器運作流程圖

四、RSS處理器

當障礙通告訊息經過了障礙政策制訂模組以及障礙萃取器兩個模組的過濾與萃取後，傳到RSS處理器中的資料即為迫切急需立即發佈的障礙通告。在RSS處理器中，我們會事先建置了以XML Schema所定義的RSS 模版，藉由XML的XSLT格式轉換功能，依照事先已定義好的XML Schema，將資料匯出成為一個RSS規範的新聞發佈檔，即時通知相關人員(訂閱者)進行障礙排解動作。在RSS檔案中，其主要核心稱為“項目 (Items)”，而在每一個項目中，至少包含了三項資訊，分別為：“標題(Title)”、“說明(Description)”以及“連結(Link)”，本系統所制定的RSS 模版，其詳細的規格定義如圖14 所示。

```
<?xml version="1.0" ?>
<rss version="2.0">
<channel>
    <item>
        <title>[Network Device] 障礙通告</title>
        <description>
                障礙發生時間: [Time]
                障礙發生地點: [Location]
                障礙描述: [Fault Description]
                建議處置說明: [Suggestion]
        </description>
        <link>http:// </link>
    </item>
</channel>
</rss>
```

圖14：RSS 模版

RFNS利用RSS的訊息即時發佈能力，建立快速障礙通告發佈機制；利用整合的功能，讓網管人員只要在將單一管理介面中，就能瀏覽掌控所轄網路設備之障礙通告狀態；利用訂閱的功能，讓網管人員在分散式的網路管理架構下，訂閱所負責網路區段的障礙通告發佈頻道(Channel)；採用標準的HTTP協定，可執行於Web-based的環境。

肆、績效模擬及結果分析

一、模擬環境設計

在本章節中我們將運用上一章節的架構使用JAVA開發工具實際建置RFNS，並設計相關情境進行模擬實驗，圖15為RFNS與XML-based網管環境之間的關聯圖，主要是將本論文所提出之RFNS架設於SNMP/XML gateway與網路管理人員之間，主動接收經由SNMP/XML gateway產生之XML格式之MIB資訊檔，依據RFNS所設定之障礙政策，過濾出符合通告的障礙資訊，再藉由障礙萃取器排除掉已經通報過但尚未解除的障礙事件，進一步精萃出迫切急需處置的網管事件，再將這些事件轉成RSS格式的障礙新聞，即時通報給網管人員知悉，好讓網管人員能夠盡速進行處置動作。

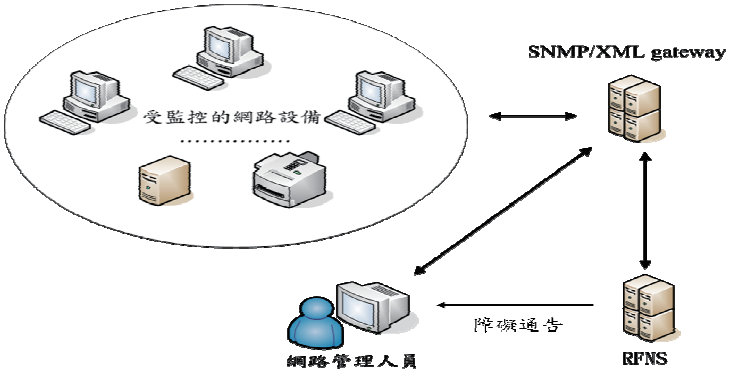

圖15：RFNS與XML-based網管系統關聯圖

在模擬實驗的過程中，我們將利用MRTG(Multi Router Traffic Grapher)流量工具，取得下列四項關鍵績效指標(Key Performance Indicators, KPIs)，藉以證明使用本論文所提出的RFNS系統，確實能夠大幅降低XML-based網管系統在障礙通告服務上所佔用的網路頻寬，有效精萃了障礙通告的檔案內容，更進一步的大幅縮短了障礙處理時間，提昇障礙管理的品質。這四項關鍵績效指標分別為：(1)網路輸出(Out)流量(2)CPU負載率(3)記憶體使用量(4)系統產出之障礙通告檔案大小。其中「障礙通告檔案大小」，指的是通知網管人員的障礙通告內容，在有執行RFNS的網路環境中此檔案為經過萃取過濾後，傳給網管人員之RSS文件檔；而在未執行RFNS的網路環境中，此檔案為SNMP/XML gateway所產生的XML格式之MIB資訊文件檔。

接下來介紹本文模擬實驗所規劃的軟硬體環境與相關情境內容的設計，首先針對軟硬體環境的建置進行說明，如圖16所示，我們規劃了三個不同的組件(components)，分別為(1)SNMP/XML gateway的模擬伺服器：做為儲存被控管所有網路設備之XML 格式的MIB資訊文件檔，並授權經由RFNS模擬伺服器定期讀取所儲存之XML文件檔之要求；(2)RFNS系統的伺服器，以五分鐘為固定週期，定期自動讀取由SNMP/XML gateway產生的XML格式之MIB資訊文件檔，經由RFNS系統的過濾萃取所需的障礙事件，產生RSS格式之障礙通告檔，再經由HTTP的協定，將重大的障礙資訊，即時的傳送給網管人員；(3)接收障礙通告的網路管理工作站。依據POSTECH團隊所定義的MIB資訊庫之XML Schema（Yoon et al. 2003），我們製作了500個XML格式之MIB資訊檔，存放在SNMP/XML gateway的儲存裝置中，代表在網路上的500個網路設備，並設計25種的預警及網路障礙事件，包括了網路介面裝置流入的位元流量超過最大可容許的最大上限(使用ifInOctets物件)、網路介面裝置無法正常作業(使用ifOperStatus物件)、因傳輸產生錯誤所丟棄的封包數量超過可容許的最大上限(使用ifInErrors以及 ifOutErrors物件)、設備完成接收所有片段封包後但無法正常重組的數量超過可容許的最大上限(使用ifReasmFails物件)、收到ICMP封包超過可容許的最大上限(使用icmpInMsgs物件)、無法ping到網路設備的錯誤數量超過可容許的最大上限(使用icmpInDestUnreachs物件)、時間超過的ICMP的控制訊息數目超過可容許的最大上限(使用icmpInTimeExcds物件) 、目前已建立的TCP連線數超過可容許的最大上限(使用tcpActiveOpens物件)…等，並將這些狀態設計在相關的MIB物件欄位中，同時我們也在RFNS的FNCD資料庫中定義了上述25項預警及網路障礙事件的通告政策記錄。

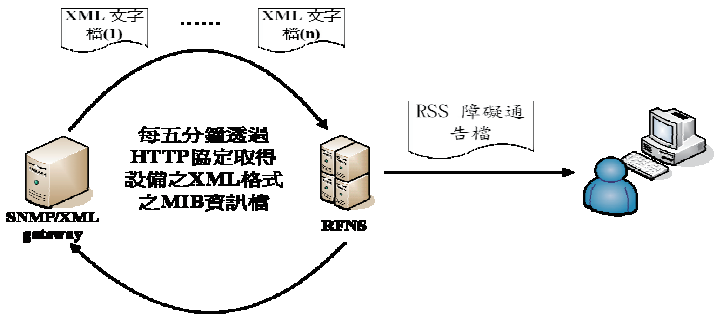

圖16：模擬實驗所建置的環境說明圖

模擬過程說明如下：每五分鐘RFNS會按照事先所定義之各設備讀取MIB資訊之HTTP Get指令，循序執行，依次抓取各設備所專屬之XML格式之MIB資訊檔，接著再匯入障礙政策處理模組，過濾出需要發佈障礙通告的網路設備與相關的物件內容，並從FNCD中讀取網管人員應進行處置的標準流程，再將相關資訊傳送至障礙萃取器中，萃取出尚未發佈的障礙事件及其相關資訊，接著再將這些資訊傳送至RSS處理模組，將預備發佈之障礙通告相關事項轉製成RSS新聞通告文字檔。而網管工作站也以每五分鐘的固定週期，讀取由RFNS所產生之最新狀態的RSS格式之障礙通告新聞。另外我們知道當網路某些節點發生重大故障時，常常會發生嚴重的連鎖反應。所以在此實驗中，我們將模擬上述狀況，在第一個五分鐘內有五十個節點發生故障，而後再以每五分鐘為一個單位，每經過一個單位便會增加五十個節點發生障礙，且假設之前發生障礙之節點尚未回復正常狀態，並連續進行十個單位。

在XML-based網管環境中，雖然有提供透過HTTP Get指令取得單一網路設備之所有XML格式之MIB資訊，但並沒有提供製作障礙通告檔案的相關訊息（Bourges-Waldegg and Hoertnagl 2005; Choi et al. 2003; Klie and Straus 2004），因此在本模擬實驗中，我們將以SNMP/XML gateway所產生的原始XML格式之MIB資訊檔案內容做為未執行RFNS服務所產生之障礙通告檔案，因為在實際的狀況下，XML-based的網管人員必需經由HTTP Get指令才能取得單一網路設備的所有MIB資訊，也必需經由該XML內容的解讀，才能進一步的進行障礙偵測工作。

另外，本模擬實驗透過MRTG(Multi Router Traffic Grapher)工具取得相關KPI數據。MRTG是目前最多人使用的流量統計工具，它可以利用SNMP協定向網路設備或伺服器取得相關的數據資料，並自動依據該資料繪製出統計圖表，使用者可以自行編輯「MRTG設備組態檔」，設定要監控那些系統或網路資源。經由上述模擬實驗情境之建置，然後進行模擬並實驗而獲得執行RFNS與未執行RFNS的四個KPI相關數據值，其詳細結果如表3所示。

表3：相關KPI之實驗結果彙總表

	平均輸出流量 (Byte/sec)		平均CPU使用率		平均記憶體耗用量 (Mbyte)		產出檔案大小 (Kbyte)	
	無PFNS	執行PFNS	無PFNS	執行PFNS	無PFNS	執行PFNS	無PFNS	執行PFNS
0～5分鐘	159	282	0%	1.00%	167.9	505.6	59	8
5～10分鐘	716	650	1.00%	2.00%	266.7	409.7	118	23
10～15分鐘	1241	972	1.00%	3.00%	316.7	433.6	177	19
15～20分鐘	1775	1143	1.00%	3.00%	326.8	509.6	236	23
20～25分鐘	2310	1476	1.00%	3.00%	333.6	529.9	295	27
25～30分鐘	2829	1812	1.00%	4.00%	338.6	529.8	301	31
30～35分鐘	3372	2144	1.00%	5.00%	342.4	529.9	413	34
35～40分鐘	3891	2478	1.00%	6.00%	345.5	529.9	60	39
40～45分鐘	4421	2814	1.00%	8.00%	348.1	529.9	530	43
45～50分鐘	4957	3149	1.00%	9.00%	350.3	530.4	589	47

二、績效評估及結果分析

經由模擬實驗產生的KPI數據加以分析，我們發現在有使用RFNS服務的狀態下，其CPU及記憶體的使用量上呈現較高的需求，從表3可以看出當有500個網路節點同時發生障礙事件或所預警的狀況出現時，CPU的平均耗用率微升到9%，而記憶體也多出了200M左右的使用量，增加的主要原因在於系統需先將網路設備所傳回之相關訊息進行過濾及萃取的動作，故必須使用到較多的系統資源，但此種狀態可以經由硬體設備的升級，以及其他擴充資源的方式加以改善。

但在網路的頻寬使用量上卻可得到大幅的改善，在圖17的輸出流量比較統計圖中我們發現在有使用RFNS服務的狀態下，當網路出現100個節點故障時，其流量較未使用RFNS有9%的減少率，而隨著網路障礙節點數的增加，網路流量的減少率也不斷的提升，到了網路障礙節點數到達500時，網路流量的佔有率已較未使用RFNS的狀態減少了37%，依據統計圖的趨勢推論，當網路節點數量越多時，RFNS服務所提供的頻寬節省效率也會越佳。當網路環境發生重大障礙事故時，可能會引發極其嚴重的連鎖反應，例如：嚴重的封包碰撞(collision)造成網路的擁塞；或是網路頻寬遭眾多的障礙相關資訊流(如：trap、alarm..等)所佔據，此時網路可使用的頻寬勢必成為極其重要的資源，因此RFNS經由障礙萃取器及障礙政策制訂模組兩項模組，降低頻寬的耗用量，便成為極其有效率的障礙通告功能。

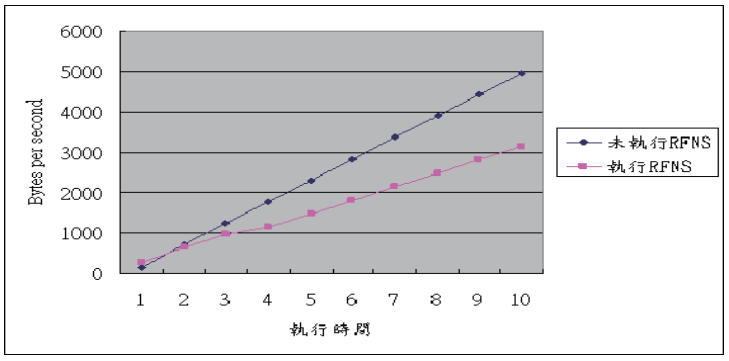

圖17：輸出流量比較統計圖

障礙通告檔的內容愈精簡，象徵網管人員愈能夠快速找到障礙事故發生的原因，進而愈快能夠將障礙排除，甚至能夠預防重大障礙的發生。由表3我們可以看出有使用RFNS服務所產生的障礙通告檔在有50個節點發生障礙事件時，其產生的障礙通告檔大小為未使用RFNS服務產生的檔案大小的13%，而隨著網路障礙節點數的增加，未使用RFNS的狀態所產生的檔案大小呈現倍數成長，而使用RFNS服務的狀態其所產生的檔案大小則都維持在50K以下，當網路上有500個節點發生障礙事件時，未執行RFNS服務所產生的障礙通告大小是有執行RFNS狀態下的12倍之大。觀察圖18的比較統計圖我們也可以發現，隨著網路障礙節點數的不斷增加，有執行RFNS與沒有執行RFNS所產生的障礙

通告檔案大小兩者之間的差距愈拉愈大，這意味著在未執行RFNS狀態下的網管人員必需讀取比有執行RFNS服務更多內容的網管資訊，才能找出可能的障礙事件，而其所耗費的障礙偵測時間相信也絕對遠遠多於有執行RFNS服務的網管人員。

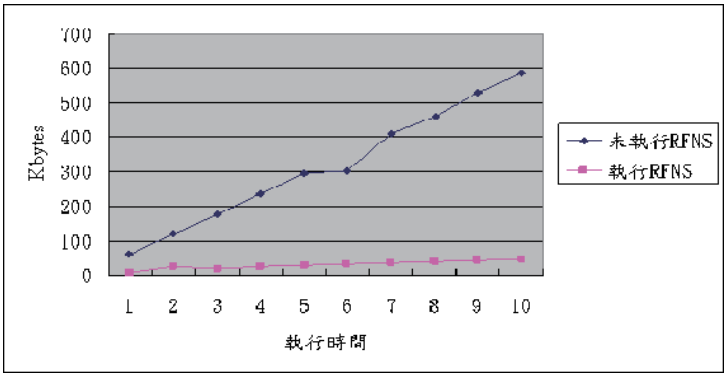

圖18：障礙通告檔案大小比較統計圖

「障礙處理時間」是另外一個非常值得探討的重要指標，我們知道在障礙處理的過程中，包含了三個主要步驟，分別為步驟一的「障礙偵測」，就是發現障礙產生的原因；緊接著為步驟二的「隔離障礙」，就是儘速將發生障礙的設備予以隔離，避免影響到其他關聯的節點，造成連鎖反應；第三步驟則為排除障礙，也就是將發生障礙的設備予以修復，讓網路能夠恢復正常狀態（ISO 7498-4），所以整體的障礙處理時間之計算公式如下：

障礙處理時間 ＝ 障礙偵測時間 ＋ 隔離障礙時間 + 排除障礙時間 … (1)

欲完成障礙排除的最終目標，就必需要能夠先發現障礙產生的原因，所以說「障礙偵測」的時間長短是決定障礙處理時間的最關鍵要素。試想若障礙偵測花費的時間過久，則勢必延後了障礙節點的隔離時間，在前面我們曾經提到，隔離障礙節點的主要原因在於避免影響其他關聯的網路節點，所以進行隔離障礙動作的時間愈晚，則造成的損壞範圍就會愈大，網管人員勢必就得要花費更多的時間在障礙排除的工作上，所以我們可以合理的推論出「障礙偵測」所耗費的時間長短，勢必會以倍數的增長，反應在整體的障礙處理時間上。表4說明在本模擬實驗中，執行RFNS服務進行障礙偵測所取得的相關系統回應時間(Response Time)之完整數據，而關於此系統回應時間指的就是障礙偵測時間。觀察「單一障礙平均偵測時間」欄位內容我們發現，在第一個五分鐘所呈現之數據值遠遠大於之後時間單位所產生的相關數據有1.85倍，其主要原因在於系統剛開始執行時，RFNS未過濾出有任何重覆的障礙事件(因為資料庫中沒有任何事件記錄)，因此系統必須將所有障礙事件之相關資訊記錄於資料庫，所以在圖19的「單一障礙平均偵測時間」統計圖中先將第一單位的數據予以排除，以呈現出較明確之整體趨勢走向。由圖19我們可發現，當障礙事件發生的數量愈來愈多時，其「單一障礙平均偵測時間」也呈現了逐漸遞減的趨勢，所以由此實驗結果證實本系統使用RFNS確實能夠縮短障礙偵測時間。尤其當出現大量未修復完成之障礙事件，不斷重覆出現在每次Polling偵測中的這種

常見狀態，其更能大幅的縮減障礙偵測時間，再經由公式(1)的導入，所以我們可合理的推論，有使用RFNS服務的XML-based的網管系統確實能夠降低障礙處理所花費的時間，進而提高了整體的障礙管理之績效。

表4：RFNS系統回應時間表(每單位時間增加50個障礙節點)

KPI \ 時間	0～5分鐘	5～10分鐘	10～15分鐘	15～20分鐘	20～25分鐘	25～30分鐘	30～35分鐘	35～40分鐘	40～45分鐘	45～50分鐘
系統整體回應時間(Sec)	15.373	16.62	24.584	32.379	40.333	48.033	55.815	63.29	70.732	78.296
障礙發生數量	150	300	450	600	750	900	1050	1200	1350	1500
單一障礙平均偵測時間(Sec)	0.1025	0.0554	0.0546	0.0540	0.0538	0.0534	0.0532	0.0527	0.0524	0.522

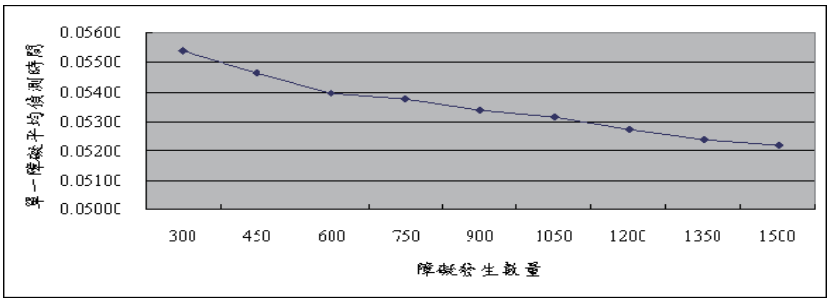

圖19：單一障礙平均偵測時間統計圖(每單位時間增加50個障礙節點)

為了觀察RFNS在每單位時間增加的障礙節點數不同，其效能是否也會有不一樣的表現，因此在本論文中另外設計了一組為每單位時間增加30個障礙節點之實驗情境，表5說明了在此實驗中，執行RFNS服務進行障礙偵測所取得的相關系統回應時間(Response Time)之完整數據，圖20則為此實驗之「單一障礙平均偵測時間」統計圖。我們發現在這個實驗中其「單一障礙平均偵測時間」較其前一個實驗之時間來得短，研判其主要原因應該是在於RFNS處理的資料量較少，所以執行時間比較快的緣故。另外我們也發現當障礙事件發生的數量愈來愈多時，其「單一障礙平均偵測時間」也同樣呈現了逐漸遞減的趨勢，且其遞減的幅度與前一個實驗也概略一致。故我們可以由上述兩個實驗得出RFNS在每單位時間增加障礙節點的數量上並不太會影響其效能的表現。

表5：RFNS系統回應時間表(每單位時間增加30個障礙節點)

KPI \ 時間	0～5分鐘	5～10分鐘	10～15分鐘	15～20分鐘	20～25分鐘	25～30分鐘	30～35分鐘	35～40分鐘	40～45分鐘	45～50分鐘
系統整體回應時間(Sec)	8.982	9.699	14.243	18.591	22.844	26.865	30.872	34.753	38.706	42.706
障礙發生數量	90	180	270	360	450	540	630	720	810	900
單一障礙平均偵測時間(Sec)	0.0998	0.0538	0.0528	0.0516	0.0508	0.0497	0.049	0.0483	1.0478	0.0475

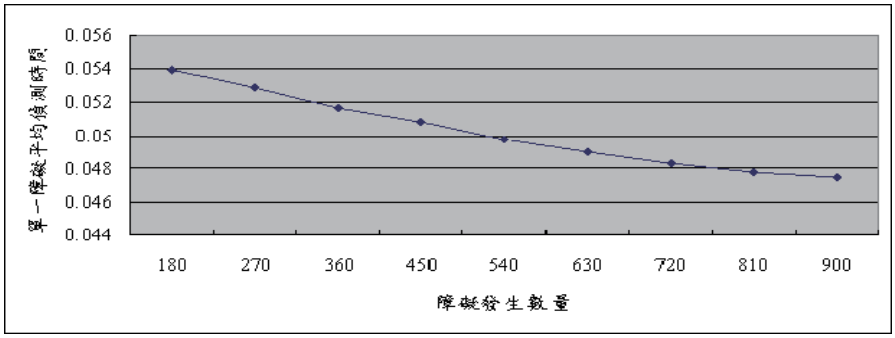

圖20：單一障礙平均偵測時間統計圖(每單位時間增加30個障礙節點)

伍、結論

由於XML-based網管系統主要在於改善因SNMP協定過於簡單，導致執行效率緩慢，並因網路設備繁多且對於各式設備MIB架構的定義又缺乏彈性等兩項屬於SNMP-based主要缺失的網管環境進行改善，以及障礙管理效率的提升。障礙管理是網路管理五大重要的功能項目之一，其中障礙的偵測及通知為整個障礙管理極為重要的前置環節，它標示著網管人員處置障礙所能運用的時間長度，並且關係著後續障礙處理的進行及品質。當網路環境發生重大障礙事故時，可能會引發極其嚴重的連鎖反應，例如：嚴重的封包碰撞(Collision)造成網路的擁塞；或是網路頻寬遭眾多的障礙相關資訊流(如：Trap、Alarm..等)所佔據，此時網路可使用的空間勢必成為極其重要的資源，因此在網路障礙事件發生時，如何能夠降低頻寬的耗用量，同時縮短障礙處理時間的長度，便成為障礙管理中非常重要的課題。

本文提出了一種新的網路障礙通告發佈系統，稱為RFNS。經由系統中的障礙政策處理模組過濾出屬於需要處理的網路障礙事件；透過障礙萃取器排除掉重覆公告的障礙通告；再藉由RSS處理模組轉換為RSS文件，即時將精萃後的障礙通告發佈給相關網管人員進行處置。同時經由模擬實驗的結果，我們發現當網路障礙節點數到達500個時，使用RFNS服務的網路流量佔有率較未使用RFNS的狀態減少了37%之多，並且依據統計圖的趨勢推論，當網路節點數量越多時，RFNS服務所提供的頻寬節省效率也會越佳，因此驗證RFNS確實能夠有效率地降低因障礙通告發佈所佔用的頻寬；同時我們也發現當網路上有500個節點發生障礙事件時，未執行RFNS服務所產生的障礙通告大小是有執行RFNS狀態下的12倍之大，因此我們也能夠證明使用RFNS確實可有效地減少因障礙通告發佈所產生的檔案大小。此外，並萃取了障礙通告的內容，讓網管人員能夠即時且正確的獲得障礙資訊。

對於「障礙處理時間」的指標，經由本文第四章的討論，我們已經證實「障礙偵測所花費時間的長短，勢必會以倍數的增長，反應在整體障礙處理時間上」。在本文的模擬實驗中，我們觀察到有使用RFNS的服務狀態下，當障礙事件發生的數量愈來愈多時，

其「單一障礙平均偵測時間」呈現逐漸遞減的趨勢，尤其當出現大量未修復完成之障礙事件，不斷重覆出現在每次Polling偵測中的這種常見狀態，其更能大幅的縮減障礙偵測時間，所以，我們可證實有使用RFNS服務之XML-based網管系統在障礙處理的時間花費上確實能夠降低，進而提高了整體的障礙管理之績效。

經由模擬實驗結果證明，本文所提出的RFNS確實能夠大幅提昇障礙管理的品質及縮短整體障礙處理時間。未來，仍有下列幾項議題可進一步加以研究：(1)目前RFNS所提出的障礙政策制訂模組，其障礙政策的過濾方式，一次只能比較MIB中單一物件是否符合政策所設定之範圍值，而無法進行複合物件的條件比較。因此未來可將FNCD的資料結構(Schema)改進成為使用MIB複合物件條件比較之政策過濾器，甚至可將其轉換成為知識庫(Knowledge Base)架構，藉由網管專家經驗法則的導入，進而提高障礙預警的效能，以防堵障礙事件發生，而非僅止於縮短障礙處理時間的目標了；(2) RFNS採用JAVA語言開發設計，不受作業系統限制。因此未來我們可以將RFNS服務移植於免費的作業系統上；(3) 目前的RFNS不論是對上游的SNMP/XML gateway，或是對發佈對象的RSS Reader，其傳輸協定皆採用HTTP，因此沒有搭配任何的網路安全管理機制。未來可增加HTTPS協定進行加密動作，以確保網路傳輸過程的安全。

誌謝

感謝兩位匿名論文審查委員惠賜寶貴的意見，使得本論文的內容更加充實與完善。

參考文獻

1. Arrowsmith, R., and Tracy, W.“Method and Apparatus for Policy-based Alarm Notification in a Distributed Network Management Environment”, *United States Patent 6057757*, May, 2000.
2. Bourges-Waldegg, D., and Hoertnagl, C.“Combination of RSS Newsfeeds and Forms for Driving Web-based Workflow”, *Proceedings of the IEEE International Conference on e-Business Engineering*, October 12-18, 2005, pp.142-149.
3. Choi, M. J., Oh J. M., and Hong, J. W.“Design and Implementation of XML-Based Management Agent”, *Proc. Of IEEE/IFIP Asia-Pacific Network Operations and Management Symposium* (APNOMS2003), Fukuoka Japan, Oct. 2003, pp. 331-342.
4. Jiang, Q., Adaikkalavan, R., and Chakravarthy, S. "NFMi: An Inter-domain Network Fault Management System", *21st International Conference on Data Engineering (ICDE'05), 2005*, pp.1036-1047
5. Ju, H. T., Choi, M. J., Han, S., Oh, Y., Yoon, J. H., Lee, H., and Hong J. W.“An Embedded Web Server Architecture for XML-based Network Management”, *Proc. of the IEEE/IFIP*

Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April, 2002, pp. 5-18.

6. Klie, T., and Straus F.“Integrating SNMP Agents with XML-based Management Systems”, *IEEE Communications Magazine* 2004, (42:7) , pp.76-83.
7. Martin-Flatin, J. P.“Web-Based Management of IP Networks and Systems”, *Ph.D. Thesis, Swiss Federal Institute of Technology* (EPFL), October 2000.
8. Rose, M. T., and McCloghrie, K.“Structure and Identification of Management Information for TCP/IP-based Internets” *RFC 1155*, May 1990.
9. Sirajuddin, S., and Sqalli, M. H.“Distributed XML-based Network Management Using JPVM”, *International Journal of Network Management* (IJNM), Vol. 16, July-August 2006, pp. 263–277.
10. Wu, J., and Yan, G.“A New Approach to Implement Enterprise Content Management System Using RSS and Folksonomy”, *Research and Practical Issues of Enterprise Information Systems II*, (255), 2008, pp.1101-1110.
11. Yoon, J. H., Ju, H. T., and Hong, J. W.“Development of SNMP-XML Translator and Gateway for XML-based Integrated Network Management”, *International Journal of Network Management* (IJNM), (13:4) , July-August 2003, pp. 259-276.
12. International Organization for Standardization, Information processing systems-Open System Interconnection-Basic Referenct Model-Part 4: Management framework, ISO/IEC 7498-4:1989 (E).
13. IETF,“Network Configuration (netconf)”,http://www.ietf.org/html.charters/netconf-charter.html.
14. W3C. Extensible Markup Language (XML) 1.0. W3C Recommendation, October 2000.
15. W3C. XML Schema Part0: Primer. W3C Recommendation, May 2001.
16. W3C. XML Schema Part1: Structures. W3C Recommendation, May 2001.
17. W3C. XML Schema Part2: Data Types. W3C Recommendation, May 2001.