以 SOAP 為基礎的物件位置管理服務之研究

葉耀明、黎俊彥
國立台灣師範大學資訊教育研究所
ymyeh@ice.ntnu.edu.tw、percylee@ice.ntnu.edu.tw

摘要

分散式物件技術在日益成熟之後，已逐漸成為運算模式的主流，而架構在 XML（Extensible Markup Language）之上的 SOAP 也成為這個環境中真正跨平台、跨語言的分散式溝通協定。然而處於分散式環境中的物件，可能因為主機故障或通訊網路中斷，而無法正確運作。為了使系統能有更高的可用度（Availability），必須在這個新協定上考慮容錯技術。

實作上，SOAP-based 平台也需要服務物件註冊與找尋機制，本論文乃提出以 Quorum-based 位置管理方法實現之，並以 SOAP Message 達成物件複製（Object Replication）。物件之間採 SOAP 為訊息溝通協定，且實現主體-備份（Primary-Backup）、動態投票（Dynamic Voting）、Quorum 等不同的容錯策略，因此本系統可稱之為 SOAP-based 容錯物件服務平台，適合作電子商務之應用。本論文中也實作了股市資訊、郵局帳號管理、聊天室等示範系統，藉以展現容錯平台之能力以及應用物件之實作方法。

關鍵字：SOAP、XML、重複物件、動態投票、Quorum、物件複製

1 簡介

今日的軟體系統可以由一系列 Loosely-coupled、Dynamic binding 的服務所構成，而服務的提供者，將會是分散在各個不同主機上的服務物件。當物件的概念實作於分散式環境中，軟體執行平台逐漸發展為分散式物件環境。

分散式物件的概念帶來了許多好處，首先可以整合不同的單位、不同的組織單元。商業上的應用系統所涉及的層面通常很廣，此時可以由各個領域的專業廠商提供服務物件，當應用系統流程中需要某服務時，再進行遠端呼叫。其次，資料與物件的距離可以更近，因為物件的概念接近獨立的個體，因此可以方便地複製到不同的主機上，當使用者端執行物件叫時，可以就本地端的物件優先選用，以實現物件快取。對整體而言，這也是一種負載分散（dissemination），可提昇系統效能。此外，分散式物件可以提高系統擴展性，例如某個資料處理物件過於忙碌時，可以單獨就該物件執行擴充，一旦提供服務的物件數目增加，資料處理時間就可以縮短。

一般來說，Component-based Software 必須要有平台的支援且透過網路才能實現，上述的優點甚至必須要在 WAN 上面跨平台才得以完成。目前有兩個廣受歡迎的分散式系統平台，亦即 DCOM（Distributed Component Object Model）與 CORBA（Common Object Request Broker Architecture），其設計上主要以 LAN 為預設環境，而且所使用的通訊協定都有特殊的規格，雖然效能高，相對在使用上也必須搭配專用的支援套件。例如：CORBA 須配合 ORB（Object Request Broker）才能運作。特有的訊息格式會增加開發者在讀取及除錯時的困難度，不同的 Endian Architecure 下，所產生的資料編組（marshal）也影響了組件的互通性。因此，尋求統一的規格有其必要性存在。

另外一個問題是目前 TCP/IP-based 網路中的嵌入式裝置，可能無法處理 DCOM 或 CORBA 這類較為複雜的物件通訊。在這些小型設備上，所擁有的記憶體相當有限，必須選擇一個簡單的處理模式。使用 SOAP（Simple Object Access Protocol）[3] 的呼叫只需要最基本的文字處理能力，恰可符合需求。

SOAP 所依賴的底層通訊協定並沒有一定的限制，可以結合現有的通訊協定，對於開發平台的選擇有最大的彈性。目前大部份的 SOAP 實作都利用

HTTP 來完成，但在一些特殊的需求下，例如 HTTP 不適用處理一對多的通訊模式時，也可以利用 SMTP 來實現。

以往在 CORBA 平台上，曾有諸位學者投入容錯物件服務的研究，OMG 也在 2000 年四月，公佈 CORBA 的容錯標準規格 1.0 版。以 SOAP 為基礎的分散式物件技術是近年來隨著 XML（Extensible Markup Language）興起的概念，相關規範仍在討論之中。對於此分散式物件平台上之容錯相關問題，本論文提出一套相對的解決方案以及實作方法。

2 相關研究

2.1 分散式容錯策略

在本論文著重在如何利用複製技術將錯誤遮罩，使系統維持正常運作，並不討論 check-point 及回復（roll-back）的問題。也就是說，本論文的目的在於以 replication 為方法，使物件服務能夠具有 resiliency 的特性。由以往學者的研究中可得知，分散式系統對於重複資源，有很多種複本控制演算法，其目的在於保持資源的單一複本序列規範（one-copy serializability），以達成多個複本間的 consistency。

2.1.1 Primary-Backup

假設一個系統在 k 個節點故障之後，還能夠維持正常運作，就稱為 k-resilient。在此策略下，要達到 k-resilient 則須要有 k+1 個節點。其中，有一個節點會扮演 Primary 角色，其餘則扮演 Backup 角色。邏輯結構上，這些節點可以排列成線形，且將 Primary 置於最前端。當客戶端發出請求時，會被送往 Primary 處理，即使客戶端向 Backup node 提出請求，也必須被重導至 Primary 節點。

在 k-resiliency 的系統中，若有 k 個以上的節點故障，系統必定無法繼續運作，反之，則能將錯誤予以遮罩。錯誤的節點若恰為 Backup 角色，則系統運作完全不受影響，使用者仍然從 Primary node 處得到回應。但若 Primary node 發生故障，則必須先選出新的 Primary node 才能繼續運作。選擇 Primary node 的方式有很多種，最簡便的方式是利用其線形的結構，依序遞補。

對於通訊故障造成的網路分割情形，很明顯地，只有擁有 Primary node 的群組可以繼續運作，而其它的群組必須處於等待模式，直到故障排除且與 Primary node 重新聚合之後，才能運作。

正因為此策略對於節點故障與網路分割所採取的處理方式不同，所以能夠容錯的條件是：必須能夠區分節點故障與通訊網路分割的差異。在分散式的環境下，網路分割或者是主機故障所造成的現象皆為遠端資源無法存取，如果貿然將暫時失去聯繫的 Primary node 視為故障，而選出新的替代者，待網路重新連線之後，系統中會有兩個 Primary node，這會造成運作上的困擾。

2.1.2 Dynamic Voting

顧名思義，voting 代表對於某一個複本執行的操作都必須通過所有複本的投票決議。以參與投票的複本成員來區分，有靜態方法(static methods)與動態方法(dynamic methods)兩個類別。所謂的靜態是指參與投票的成員數目固定，而動態就可以使得參與的成員隨著系統狀態而改變。在容錯系統中，節點一旦發生故障，就會在系統中消失，待修復後又會重新加入，動態投票的方式所具有的彈性，才能適應這樣的變化。

曾經有 Sushi Jajodia 與 David Mutchler[6]兩位學者針對重複資料庫（replicate database）提出動態投票演算法，作為複本控制之用。其目的在於維持各個重複資料庫的內容一致性，並提高資料庫的可靠度。而投票動作實際上是由發起者收集各個資料庫的狀態資料，並加以統計，若符合判斷條件則允許資料更新。

動態投票演算法可使得位於多數部份的各複本間狀態一致，而多數群組只會有一個，網路分割的問題自然可以解決。然而，對於節點故障，也可以用同樣的方式達成容錯，只要在兩次投票之間，故障的節點數不超過總節點數的一半，則故障的節點為少數群組，剩餘的正常節點總數仍為多數。

這個的作法雖較 Primary-Backup 複雜，但是對

於節點故障與網路分割所採取的處理方式是完全一致的，因而不必區分其間的差異，先前提及 Primary-Backup 在容錯上的限制在此便可迎刃而解。

2.1.3 Quorum

與上述兩種策略相較之下 Quorum 有兩個不同處，首先因為這是一種 Legion Structure，因此在一致性處理上，不需對所有複本進行同步，而只針對選定的集合（Quorum Set）來處理。其次是各複本的操作方式是由客戶端獨立呼叫，不但個別有獨立的結果，且不需要複本間訊息傳遞。

此策略的特點為，每一次的資料更新都會保證該更新集合內的複本狀態是最新且一致的。對於集合外其它複本的資料不一致以及過時資料都可以利用時間戳記來排除，不影響運作。

本文所採用的 Quorum 策略的各複本排列方式並不需要特殊形態，但為了 Quorum Set 選取及說明的方便，邏輯上可將其排列為環狀。若有系統中有 N 個複本，則索引值為 0 至 N-1。進行 Quorum Set 的選取前，首先會在這 N 個索引中隨機找出一個起始點 i，再求出整個集合。

在容錯方面，上述之變數 i 是隨機產生的值，這也意味著每次選取的集合都不完全相同。以 U-Set 來說，每次更新的值會置於不同的複本之中，若發現所選取集合之中有某個複本在更新後無法回應，則可隨機再選出新的 U-Set 作為替代，保證資料更新能完全正確。而隨機選取 Q-Set 與 U-Set 必定有交集，若交集處正好發生故障，也可以隨機再選出新的 Q-Set、再產生新的交集，確保查詢能夠順利完成。

Quorum-based 演算法的對稱性（symmetric），使得每一個複本在不同的集合內，出現的次數是均等的。再配合上述隨機選取的作法，此策略會比先前的討論多了負載平衡（load balance）的優點。

2.2 Loosely-coupled 分散式物件通訊協定: SOAP

SOAP 繼承了 XML 的優點，其原本的用途是在分散式電腦環境中作資料共享。起初由 DevelopMentor、UserLand Software 及微軟公司提出。隨後 IBM、Lotus、惠普、昇陽等公司也全力支持，這使得 SOAP 演變成網路服務的重要技術標準。因為 SOAP 的核心是 XML，故用 XML Schema 來規範訊息中的資料型態，除此之外，再以 Element 和 Envelope 來做編碼及包裝。雖然 SOAP 是以單向的訊息傳遞為基礎，但可以結合兩次訊息傳遞，形成 request/response 的模式。換句話說，物件透過 SOAP 傳訊息也就等於互相交換 XML 格式的文件。

SOAP 在定義上可以分為四個主要的部份，Envelope 如同真實信封一樣，包含了 SOAP Message 的發送者、接收者、訊息的內容以及如何處理該訊息等資訊。而 Encoding 部份則定義 SOAP 訊息的編碼方式，以及如何利用 XML Schema 描述使用者自定的資料型態。至於 RPC 的部份在規定 SOAP 在遠端程序呼叫的應用，亦即如何在訊息之中包含目地物件的 URI、呼叫介面（method）以及參數等。最後，在 Binding 方面定義了 SOAP 與其底層的通訊協定間的關係，如與 HTTP 的結合。

以往 DCOM 只能用於 Windows 系統，RMI 只能用於 Java 環境，CORBA 對語言及平台雖無特定的限制，但是每個節點都要有 ORB 的支援才能運作。與其他的物件通訊協定相較之下，SOAP 是真正開放的架構，其訊息僅是簡單的 XML 文件，沒有平台更沒有語言的限制。

3 分散式容錯物件服務

本論文在於探討 XML 平台上以 SOAP 進行物件容錯、複製的意義及相關問題，並提出解決問題的對策。系統運作概念如圖 1 所示，主機間的通訊協定為 SOAP，客戶端可經由 Apache SOAP 或 JAXM[8]函式庫產生 SOAP 訊息，以呼叫服務物件。而伺服端的物件間互相溝通、使用者端對服務物件之查詢、SOAP Server 的位置登錄及更新也都以 SOAP 訊息達成。

圖 1 中的 Object Manager、Fault Tolerant Object（FTO）、Object Observer 及 Domain Server 是自行

開發的成果，而 SOAP Server 則採用 Apache 提供的套件。對於容錯服務應用的開發者來說，只要繼承 FTO，就可使其物件具有容錯能力。因此可以將大部份的時間投入在物件服務本身的設計上，使得程式開發的流程更快速，減少除錯的時間。

結構上位於核心的 Domain Server 是一個 Quorum-based 位置管理系統。由於本論文需要一個 Naming 機制，又因為 SOAP Server 所在的主機可能是一個行動裝置，其位置可能隨時會改變，再加上容錯的考量，為了讓服務可以隨時讓客戶端可以使用，因此提出一個特別設計的物件找尋的機制。在本實作中，Domain Server 利用 Sun 提供的 JAXM 套件所開發，其功能在於提供物件的名稱與位址的對應，與物件本身的容錯能力無關。因此對於容錯物件服務平台來說，可以看成是一個獨立的外部輔助系統。換句話說，如果有其他機制可以取代其功能，Domain Server 可以依需求更換，而不影響容錯物件的功能。

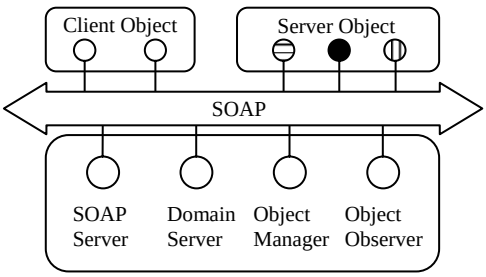

圖 1 容錯物件平台

3.1 分散式容錯物件結構

依照容錯策略的不同，在實作上會有各種不同的容錯物件產生，因此可以將其共同的部份抽出，形成階層式的結構。如圖 2 所示，本論文以重複物件為核心概念，所有的物件都會繼承 Replicate Object，再依照不同的容錯策略所需，衍生出不同種類的物件。

3.1.1 Primary-Backup 物件

這是一種最基本、且最直觀的容錯方法，正常情況下，當使用者發出服務請求時，會先經由 Domain Server 查詢服務得到一個 Primary 物件的位址，再進行物件呼叫。Primary 物件在第一次接受呼叫時，也會利用 Domain Server 查詢所有 Backup 物件的位址，並儲存於快取之中，以便將來的呼叫可以隨時取用。接著利用 SOAP 將呼叫送給所有 backup 物件，當所有 backup 物件都接受之後，Primary 物件執行此呼叫，並將結果回覆給使用者。

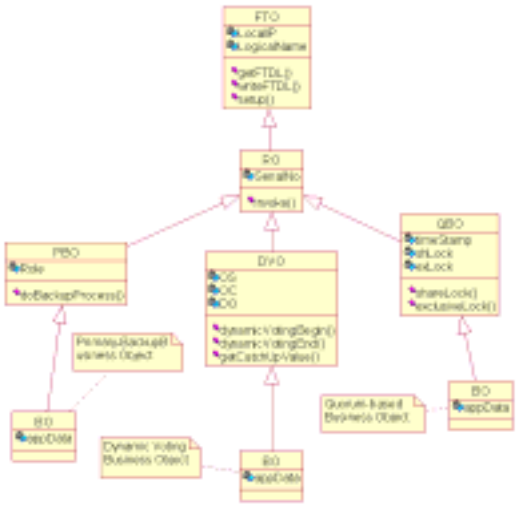

圖 2 容錯物件階層

當 Primary 物件發生故障時，經由 Domain Server 所得到的物件皆扮演 backup 角色。此時必須進行新的 Primary 物件選取。由於先前的討論得知，於此策略之下，物件可以排列為一直線。實際應用上，利用物件在 Domain Server 內註冊的順序排列，恰可以得到邏輯上的線形。一旦在 Primary 物件無法提供服務時，即假設其已經發生故障，接著找到下一個可正常工作的物件，取代成為新的 primary 物件，使得提供的服務不致於停擺而達到容錯的目的。至於如何判斷物件故障或是網路分割造成的暫時性通訊中斷，則不在討論範圍之內。

3.1.2 Dynamic Voting 物件

為了使得本論文內每一個投票物件皆允許使用者呼叫，本論文也利用動態投票演算法實作容錯物件。任何一個物件在接受使用者的請求之後，都可成為投票的發起者，接著從投票的結果中，判斷目前物件所在的群組是否為多數，位於多數群組的物件可以提供服務，其他群組的物件則必須中止服

務，以維護執行服務所需的物件一致性。

每個容錯服務物件除了必須實作投票演算法之外，在投票的機製運作上，物件內部設計有以下的輔助參數：

（1）物件狀態(Object State, OS)：為一個整數值，用來表達物件內的資料被更新的次數，也就是說每提供一次服務其值就會增加。投票機制再度啟動時，擁有最新狀態的物件才有投票權。

（2）物件基數(Object Cardinality, OC)：為一整數值，表示目前群組中，擁有最新狀態的物件個數。同樣地，每次提供服務後會重新計算，作為下次投票的參考值。

（3）識別物件(Distinguished Object, DO)：為一個物件的識別碼，在本實作中，該識別碼以主機位址、物件邏輯名稱及序號所組成。因此，每個物件都有唯一的識別碼。

3.1.3 Quorum 物件

無論是 Primary-Backup 或是 Dynamic Voting 策略在物件狀態的更新上，都必須針對所有物件作處理。投票物件甚至在提供查詢服務時，也必須經由所有成員的投票才能保證該物件的值為最新狀態。如果物件的總數增加，或者通訊網路的頻寬不足時，物件之間通訊的次數更值得關切。若以 Quorum 的方法選出特定物件，當物件提供服務時，只有被選定的物件必須參與運作，則可減少通訊次數。

圖 3 中有二十個節點，代表經複製產生的物件。當客戶端發出 Update 請求時，隨機產生的亂數為 1，此時所產生的 Update-Set(1)為{ 1 , 2 , 6 , 8 , 11 , 14 , 16 , 20 }，如圖中的灰色節點所示。依照請求執行更新動作後，集合中的每個節點都會擁有最新的狀態。接著若使用者提出查詢請求，且隨機產生的亂數為 4，則生成的 Query-Set(4)為{ 4 , 9 , 14 , 19 }。其中節點 14 為兩個集合的交集處，由此節點可以得到最新的狀態值。同理，若由 13 為起始點產生查詢集合，所得到的 Query-Set(13)為{ 13 , 18 , 3 , 8}，其中節點 8 與 Update-Set(1)也有交集，仍然可以得知物件最新狀態。

Quorum Scheme[13]在容錯上，有其獨特的表現方式，也就是利用隨機產生的亂數，取得不同的 Quorum Set。以前例來說，利用圖 3 之 Query-Set(4)執行查詢時，若發現該集合中任何一節點沒有回應，隨機產生的 Query-Set(13)可用來替換。同理，在 Update-Set 的容錯處理上，也可依相同的模式進行。

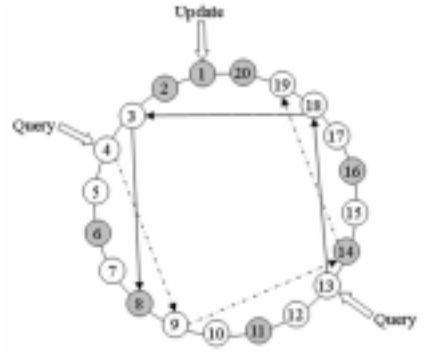

圖 3 Quorum-based 容錯策略操作範例

3.2 SOAP-based 容錯物件平台

「重複」在分散式系統的容錯設計上，是常見而有效的方法。在物件服務的設計上亦是相同的概念，若將一個物件的複本分別複製到不同的主機上，再加以有效管理，就可以在某個物件無法工作時，讓系統一如往常地提供服務，客戶端甚至察覺不到伺服器端的異狀，只是感受到運作時間拉長，但結果仍然正確。

本論文即以物件複製為核心概念，實現分散式物件服務架構。如此一來，物件服務不再因單一物件的失效而停擺。然而，以 SOAP 來進行分散式物件服務的容錯系統設計，在實作上仍有一些問題須加以解決：

（1）物件如何包裝在 SOAP Envelope 之中，加以傳送並啟動

（2）服務物件的位置之登錄與查詢

（3）容錯物件狀態的描述

3.2.1 物件複製

欲將容錯物件分散至各個主機上，可以利用網路檔案傳輸的方式分別傳送，再連線至各主機將物

件一一啟動，但這就顯得相當麻煩而且缺乏效率。若利用 SOAP 來實作，由於其僅為 XML-Based 通訊協定，還必須讓物件包裝在 Envelope 之中，再利用適當的輔助機制，才能夠自動複製至所有目的主機。

為了讓物件的複製工作可以更簡便，在這個系統中，設計了一個圖形界面的物件管理員程式(Object Manager, OM)，利用這個工具，系統管理者可以選定被複製的物件以及複製的目的地位址，並啟動複製功能。當 OM 進行複製工作時，所使用的方法就是 SPAP 附加檔，亦即 SOAP Messages with Attachments[1]的規格。

一個 SOAP 的訊息包括 Primary MIME Part 也就是一般所謂的 Envelope 部份，以及 Attachment 的部份。複製的過程中所需要的資訊有三項，分別是：

（1）Apache SOAP Server 的物件啟動資訊。當物件在 SOAP Server 中被 deploy 成為一個服務時，所需的描述資訊，其內容包括物件的邏輯名稱、CLASSPATH 以及可供呼叫的介面名稱等資訊。

（2）所有參與複製工作的 SOAP Server 列表，OM 整理出這份列表之後，會依次傳遞第一個 SOAP Server，並依複製規則傳遞下去。

（3）物件檔案及物件狀態，也就是將物件檔案編碼後的結果以及 FTDL。

在個三個項目中，(1) 與 (2) 會被置於 Envelope 之中，而（3）的部份，會附加在 Envelope 之後。

實際的複製流程是模擬細胞分列的動作，每經過一次複製程序，物件的數目會成為原來的兩倍。站台接受到複製物件的 SOAP 訊息時，會將其內含的物件還原成實體檔案，接著分析訊息內所夾帶的目的站台資訊，並嘗試啟動物件。

3.2.2 物件位置管理者(Domain Server)

本論文之實作之中，所有的物件服務呼叫，都要仰賴 SOAP Server 負責代收訊息，再依訊息目的地名稱轉發給底下的服務物件。而每個服務物件也都必須註冊至 SOAP Server 中，才能正常工作。因此，SOAP Server 的位址與系統的運作息息相關，如何管理這個重要資訊，是一個必須處理的問題。

基於容錯的考量，本論文採用 Quorum-based 分散式管理方式[14]，將資訊分散在不同的主機上。而主機位址管理系統對本容錯平台來說，是一個獨立的機制。

圖 4 選定 LS0 及 LS1 作 SOAP Server 位置更新

在負載平衡方面，此管理方式是利用隨機的方式選取 Quorum Set 來達成，如圖 4 所示。當 SOAP Server 的位址要登錄時，所選取的 U-Set 每次都不同，因此不會造成單一站台的負載過重。同樣的，當客戶端要查詢 SOAP Server 的位址時，所用的 Q-Set 也是隨機選取。利用演算法的設計，使得這兩個隨機產生的集合一定能夠產生交集。在容錯方面，當集合中的成員有任何一個發生故障時，可以隨機再產生新的集合，直到服務完成為止。

參與運作的 SOAP Server 位址如果有異動，必須向 Domain Server 登記最新的狀態。其中登記的資訊包括 SOAP Server 的識別碼(id)，時間戳記(ts)，以及最新的位址(loc)。

3.2.3 容錯描述語言(FTDL)與物件觀察代理者

概念中的物件在實作上包含物件程式碼與物件狀態值兩個部份，一般來說，程式碼在開發完成之後，就不輕易改變，而狀態則隨著物件的生命週期而變動。對應於本論文的實作上，程式碼就是 Java Class，而物件的狀態，則必須定義一個方法加以描述。

在 Java 的規格中，有所謂的物件序列化的標準

定義，使得物件可以擁有 persistence 的特性，也提供程式開發者一個將物件存檔、還原的的機制。一個物件只要實作 Serializable 的介面，便可以很容易地將內部狀擷態取並保存下來，轉作其他應用。但在這個規格中，並未定義如何以 XML 格式來作為物件序列化的內容，以致於實作出的物件序列化結果僅限於提供物件重建之用。於是有些學者將 Java 這方面的規格予以延伸，Java Serialization for XML [10] 就是一個實例。

本論文所提及的容錯物件，其內部狀態並不一定要完整的序列化，僅須錯對與容錯能力相關的部份加以整理，即可滿足需求，因此未套用上述序列化的功能，而自行定義一個 XML-Based 格式的表示方法 — FTDL(Fault Tolerant Description Language,)，做為物件狀態的描述，如下列所示。

```
<BusinessObject>
    <balance>500</balance>
    <DynamicVotingObject>
    <OS>5</OS>
    <OC>3</OC>
    <DO>http://140.122.219.57:8080/rpcrouter/
   urn:PostOffice:1
    </DO>
    <ReplicatedObject>
        <FaultTolerantObject>
            <LocalIP>140.122.219.57</LocalIP>
        <LogicalName>
            urn:PostOffice
        </LogicalName>
        <SerialNo>1</SerialNo>
        </FaultTolerantObject>
    </ReplicatedObject>
    </DynamicVotingObject>
</BusinessObject>
```

利用 FTDL 可以隨時將物件狀態記錄在檔案中，並伴隨物件檔案一同複製至目的站台，以便進行新服務物件的初始化工作。除此之外，每個物件都具有產生 FTDL 的能力，因只要呼叫所有物件所提供的相對服務，再將回傳的結果加以整理，物件狀態列表就可以很容易地得到。

3.3 客戶端物件階層

客戶端的程式會透過 SOAP Message 來呼叫伺服端的服務，所以也必須要有收發 SOAP Message 的能力，於是在本實作中，提供了客戶端的基礎物件，以便客戶端的應用程式可以快速開發。

客戶端會透過一個 Front End 與服務物件連結。實作上，本系統以 Handler 類別實現這個功能。圖 5 是 Handler 的 Class Diagram。由於不同的伺服端物件會有不同的操作方式，因此會有不同的 Handler 產生。而由 QObjRummagerClient 判別目前使用中的容錯策略，啟用正確的 Handler。

PBHandler 可以與 PBO 溝通，其功能如下：

（1）查詢並製作所有 Primary-Backup 物件列表，如果查詢成功，則於列表中找出 Primary 物件。

圖 5 客戶端物件階層圖

（2）若 Primary 存在，則利用 Replica Utilities 中的 Spokesman 向 Primary 物件請求服務執，否則選取新的 Primary 物件，再進行服務請求。

DVHandler 負責與 DVO 物件溝通，其功能如下：

（1）查詢所有 Dynamic Voting 物件，依照物件所在的位置作為選取條件，選出最適合提供服務者。

（2）對該服務物件提出服務請求。

QRMHandler 負責與 QRM 物件溝通，其功能如下：

（1）利用 Replica Utilities 中的 SetManager 產生對應的 Quorum Set。

（2）對 Quorum Set 內的物件提出請求，並蒐集回覆之結果。

進行請求結果之比對，以時間戳記最後者為最準。

ODHandler 負責與一般的物件溝通，其功能如下：

（1）查詢所有 Ordinary 物件，並依照在的位置作

為選取條件，選出最適合提供服務者。對該服務物件提出服務請求。

4 系統評估

本章將針對物件複製的效能作測試並嘗試實作應用物件。

4.1 系統環境及複製效能實測

首先在不同環境下，測試物件的容量大小與複製所花費時間之間的關係。

4.1.1 單一複本之複製時間測試

為了瞭解物件容量與複製時間之間的關係，本小節將發送端 OM-GUI 與接收 OM-Strage 透過單機、LAN 以及 WAN 作連結，希望在不同的網路環境中，比較其結果，以下為三種連結方式之作法：

（1）單機測試：兩端位於同一部主機上，亦即在本地端架設 SOAP Server，觀察在不透過網路訊訊時，所需要的複製時間。

（2）區域測試：兩端利用 10BaseT 規格的乙太網路（Ethernet）相連接。

（3）廣域測試：兩端利用 Internet 相連接。

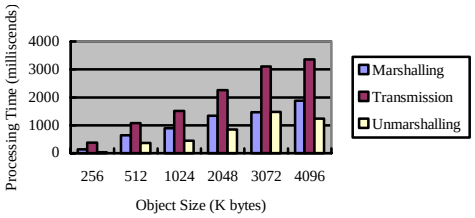

圖 6 單機測試之物件複製時間與物件資料量關係

物件傳送過種包括三個部驟，分別是物件包裝（marshalling）、利用 SOAP Message 傳遞以及將物件解包裝（unmarshalling）並存檔。以下分別就三種連結方式針對三個複製步驟各記錄其測試結果。由結果中發現，大部份的複製成本會發生在網路傳輸上，而在物件的包裝與解包裝方面，所花費的時間較為短暫，以容量為 4096K 的物件為例，包裝與解包裝所花費的平均時間約為 1.5 秒。圖 6 至圖 8 分別說明三連結方式，以 OM-GUI 發送單一物件至 OM-Stage 之結果，以物件容量與複製時間關係圖表現之。

為了降低傳輸時間，減少資訊傳遞量是一個相當直觀的方式。但在本實作中，物件皆以 Java 標準之 JAR（Java™ Archive）方式儲存，亦即格式為 ZIP 之壓縮檔。因此，對於複製之資料量無法再以壓縮方式減少，唯磁碟快取的方式仍然可行。管理者可以由物件管理員介面選擇磁碟快取功能，使得已經複製過的物件不必再次傳送，而直接取用當地磁碟中的檔案。

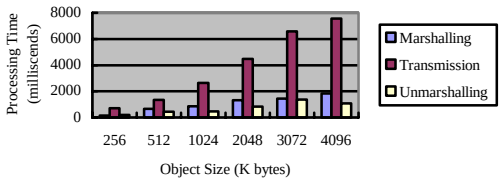

圖 7 區域測試之物件複製時間與物件資料量關係

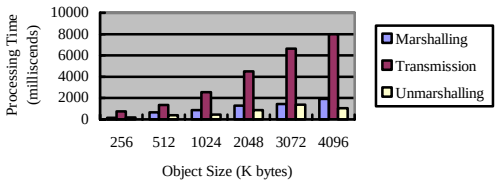

圖 8 廣域測試之物件複製時間與物件資料量關係

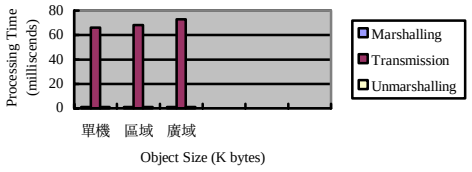

圖 9 使用快取後物件複製時間與物件資料量之關係

圖 9 表示使用快取功能後的測試結果。由此可知 marshalling 與 unmarshalling 的時間確時可以完全去除，而 transmission 的時間也因為不需物件而大幅縮減。又因為 SOAP Message 內不含物件資料，無論物件的容量大小皆不影響訊息的傳送量。

故不需討論物件容量對複製時間的影響。

4.1.2 多重複本之複製時間測試

本小節在於探討物件複本之產生與時間之關係。在測試環境方面，本小節採用的接收端 OM-Stage(B-I)皆為相同規格之機器，以便得到更準確之結果，發送與接收端以 100BaseT 之網路作連結。

由前一小節的數據可知，透過 10BaseT 的網路在兩個主機之間複製一個 4096K 的物件，完整過程大約需要 11 秒鐘的時間。由前一小節之結果可得知，大部份的複製時間決定於網路傳輸，而在本小節中由於網路速度的改進，複製時間亦大幅縮短，同樣以 4096K 的物件作測試，在 100BaseT 的網路環境下，卻只需約 4.5 秒的時間。除此之外，複製多個複本的方式是模擬細胞分裂來運作，因此產生的複本愈多時，平均的複製時間成本應該會更低。在這個測試中，也可以驗證這個假設。圖 10 將物件複製數目與所花之時間作出整理，在這個實驗環境中，當物件數目加倍時，所增加的複製時間約為 4 至 5 秒，恰為產生一個複本所需要的時間。若以此作推論，產生 64 個物件複本約需 30 秒的時間。而圖 11 顯示平均複製時間，由圖中可以看出時間成本明顯降低。

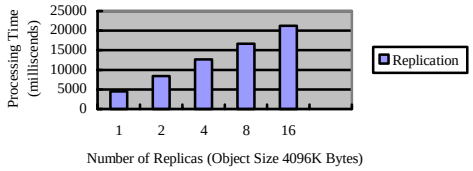

圖 10 複製複本數目與花費時間之關係

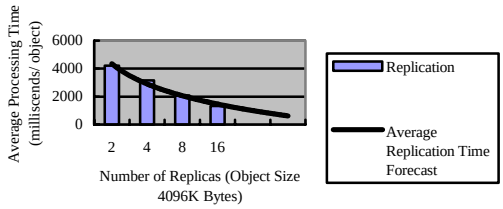

圖 11 容量為 4096K 物件複製之平均時間

4.2 應用物件與範例程式

本節利用三個程式範例分別作為三種不同的容錯策略應用實例，並利用先前開發的容錯物件服務平台進行觀察測試。

對 Primary-Backup 物件的應用，在此實作一個股市資訊服務範例，以主體物件提供股價的更新與查詢服務，並以備份物件隨著著主體物件更新狀態，目的在於使得更新與查詢請求不因服務物故障而無法運作。在使用者端可以區分為資訊的提供者與查詢者兩種角色，其中的資訊提供者通常是線上交易系統，而查詢者則是一般的交易戶。

關於 Dynamic Voting 物件的應用，在此利用提款機與郵局的互動作為範例，假設用戶操作提款機對郵局的帳戶管理主機發出請求，提供服務的主機必須有二部以上，才會有容錯機制存在。因此，在這個範例中以投票物件擔任服務主機的角色，系統運行時會同時存在多個投票物件，用戶可能對任何一個投票物件發出請求，因此物件之間必須互相溝通，以保持帳戶資料的一致性。由於動態投票演算法的特性，使得帳戶管理伺服器可以容忍主機固障及網路分割，只要有前一次提供服務的半數主機可以通聯及運作，使用者就可進行帳戶存取。而且發生錯誤的主機在修復之後，能夠自動加入運作，以回復原有之容錯能力。

在 Quorum-based 容錯物件的應用上，在先前已有 Domain Server 的實作可供參考。除此之外，本系統實作了一個聊天室程式。一般聊天室的運作方式是以中央集權的方式進行，亦即所有的參與者會連結到服務主機，並在主機上交換資訊。但在主機發生故障時，會致通訊中斷。於是在這個應用中，將中央主機程式由 Quorum-based 容錯物件擔任，並且同時有多個複本存在，但是當某個用戶發送訊息時，並非所有主機都同時進行資訊的更新，因此其他的用戶也無法由任意主機上獲得此訊息。主機選取必須遵照 3.1.3 的演算法來進行，使用者之間才能正確交換訊息。

5 結論及未來發展方向

本論文在架構上提出了一個容錯物件實作平台，以 SOAP 為核心技術，將物件複製為多個複本，並以三種不同的策略實作容錯基礎物件。利用此基礎物件及繼承的概念，可以迅速製作各種的容錯應用物件，並在本論文所建構的環境中執行，發揮容錯特性。

關於 Naming 問題的解決方面，本論文提出 Quorum-based 位置管理方式，利用行動通訊管理的概念自行製作的 Domain Server。藉此可以管理系統中所有的 SOAP Server，且實現 Portable SOAP Server 的構想，進而提供服務物件的登錄與尋找機制。此管理方式不但實作上相當簡易，且具有容錯、負載平衡等優點。

實際測試上，藉由 Java 語言跨平台的特性，使得所有開發的成果，包括登錄查詢機制、複製機制、及容錯應用物件皆不必限定在特定的作業系統之中運作，系統模組在不必重新編譯的情形下，直接在 Windows、Linux 或者其他具有 JVM 的作業系統中都可以順利執行。第五章的測試就跨越了 Microsoft Windows 與 Linux 兩個系統。

簡而言之，本論文作出以成果：

（1）容錯物件註冊與尋找機制：以 Quorum-based Domain Name Server 處理物件服務的登錄與尋找。

（2）SOAP-based 物件複製：以 SOAP Message 包裝物件，傳送至遠端並重新啟動還原。

（3）實現 SOAP-based 容錯物件服務平台：利用 Primary-Backup、Dynamic Voting、Quorum 等策略製作容錯基礎物件，完成容錯物件服務平台之建置。

在系統負載平衡方面 Lindermeier[9]在 CORBA 上曾有相關的研究，而本系統尚未討論負載平衡的處理方式。若要實作此機制，首先必須在各個 SOAP Server 所在的主機上，實作一個監控系統資源的 daemon，負責隨時蒐集伺服端的各項資源分配狀況。除此之外，必須於 SOAP Server 內實作並註冊一個負載查詢服務，收到服務請求時，可以自 daemom 處得到負載資訊，作為複製程序決定目的地的參考之用。

目前的設計上，物件雖已有階層式的劃分，但是對於各個物件的組成元素（Java Class）卻未有適當的描述，一旦物件改版時，必須將物件包裝後的 JAR 檔案整個複製至所有目地主機上，才能進行版本的更新。事實上，所謂的改版僅可能是某個類別（Class）的修訂，如果可以定義一個描述語言供物件管理員描述物件內修改過的類別，則可針對該類別進行補綴(patch)。當新版物件發行時，只要複製該語言之描述檔及相關的類別至所有目的主機，即可完成版本的更新。這樣的做法是由增值式程式碼遷移（Incremental Code Migration）的概念轉換而來，曾經有學者 Wolfgang Emmerich[5]等學者以 XML 進行程式模組的增值式修訂。當服務物件容量過大時，利用漸進式的複製方式，可以減輕複製過程所花費的成本。

面今日龐大的資訊量，以往利用瀏覽器查詢資料、下載檔案、甚至完成交易等動作，漸漸會走向電腦自動化，以代理程式來處理這些煩瑣的事務。而 Web Service 正好提供了一種簡單且順暢的機制，讓應用程式與應用程式不必經由人力的操作介入，也不必要知道兩端的環境，能自動地透過 Internet/intranet 相互溝通。Web Service 所使用的核心通訊技術亦為 SOAP，本論文中所使用之容錯策略及實作方法皆可作為探討 WebService 容錯實作之參考。

參考文獻

[1] John J. Barton, Satish Thatte and Henrik Frystyk Nielsen, “SOAP Messages with Attachments,” HTTP DOC, December 2000, http://www.w3.org/TR/SOAP-attachments

[2] Harmeet Bedi, Jonathan Chawke, Franciso Curbera, Glen Daniels, Doug Davis, Matthew J. Duftler, Kevin Mitchell, William Nagy, Scott Nichol, Sam Ruby, James Snell, Sanjiva Weerawarana, “Apache SOAP,” http://xml.apache.org/soap

[3] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer, “Simple Object Access Protocol (SOAP) 1.1,” HTTP DOC, May 2000, http://www.w3.org/TR/SOAP/

[4] George Coulouris, Jean Dollimore and Tim Kindberg, “Distributed Systems Concepts and Design,” Addison-wesley 2001.

[5] Wolfgang Emmerich, Cecilia Mascolo and Anthony Finkelstein, “Implementing incremental code migration with XML,” Proceedings of the 22nd international conference on Software engineering. 2000, Limerick, Ireland, pp.397-406.

[6] S. Jajodia and D. Mutchler, “Dynamic Voting,” ACM SIGMOD International Conference on Management of Data, pp.227-238, San Francisco, 1987.

[7] Pankaj Jalote, “Fault Tolerance in Distributed Systems,” Prentice-Hall International, Inc, 1994.

[8] “Java API for XML Messaging (JAXM),” http://java.sun.com/xml/jaxm/

[9] Markus Lindermeier, “Load Management for Distributed Object-Oriented Environments,” In 2nd International Symposium on Distributed Objects and Applications (DOA'00), pages 59-68, Antwerp, Belgium. IEEE Computer Society, 2000.

[10] Brendan Macmillan, “Java Serialization to XML,” http://www.csse.monash.edu.au/~bren/JSX/

[11] Brett McLaughlin, “JAVA and XML,” O’REILLY, June 2000.

[12] SCOTT SEELY, “SOAP: Cross Platform Web Service Development Using XML,” Prentice-Hall, Inc. 2002.

[13] Ming-Jeng Yang, Yao-Ming Yeh and Yao-Ming Chang, ”Wireless Network Location Management with Quorum-based Systems,” submitting

[14] Yeh Yao-ming, Sun Wen-Da and Chen Yeong-Sheng, “Object Replication and CORBA Fault-Tolerant Object Service,” Wu han University Journal of Natural Sciences, Vol.6, No.1-2, 2001, pp.268~277.