以寬頻網路為基礎之即時資料擷取與處理

Real-time Data Acquisition and Processing over Broadband Networks.

計畫編號：NSC 87-2219-E-002-007

計畫期限：87/5/1 - 88/7/31

主持人：王 勝 德　　台灣大學電機工程研究所

Email: sdwang@cc.ee.ntu.edu.tw　　Fax:02-2367-1909

中文摘要

隨著網路的普及，各種以網路為媒介的應用也越來越廣泛，但受限於傳統網路通信協定和現有網路頻寬的影響，使得在網路上進行即時資料處理和提供多媒體服務等應用面臨瓶頸。

本研究之目的為在寬頻網路上建立一個可進行即時資料處理的分散式計算環境，而為了簡化軟體的開發程序和減少維護所需的成本，我們採用通用物件請求仲介架構(Common Object Request Broker Architecture, CORBA)為軟體平台。

然而，在目前的 CORBA 標準規格中並未制定與即時以及容錯程序處理有關的服務，因此，在本文中我們提出一個系統化的方法來增強 CORBA 對於即時處理和容錯程序的支援；我們將在 CORBA 中嵌入支援即時處理的排程器, 以便對 CORBA 接收到的服務請求進行即時性的排程。除此之外，我們也將在 CORBA 的架構中內建可提供容錯程序服務的機制，如此，我們將可在此軟體平台上開始建構所需的分散式計算環境。

在故障停止(fail-stop)模型的假設之下，我們所提供容錯服務之偵測錯誤發生機制，的確可以偵測出應用程式發生錯誤的情況，至於在排程服務部份，目前所使用之 CORBA 系統並不支援以優先權為導向之執行機制，因此，由排程服務根據時間性要求所指派之優先權值是否合適尚有待獲得驗證。

關鍵詞：寬頻網路、容錯、即時系統、即時資料擷取、信號處理、分散式計算環境、CORBA

Abstract

In this work, we are aimed to establish a distributed computing environment based on broadband network. Furthermore, we also argue that the process and cost of developing applications could be reduced with this DCE. Thus, it is appropriate to choose the distributed object system as the development environment.

As a distributed object system, the Common Object Request Broker Architecture (CORBA) provides a well-developed solution to facilitate the development and integration of applications across heterogeneous platforms. However, with the emerging requirements for real-time and fault-tolerant support in distributed computing system. CORBA doesn't address a suitable mechanism to meet these requirements. Therefore, additional services or extensions should be added to CORBA.

In this paper, an approach that extends CORBA to support real-time and fault-tolerant requirements is proposed. First, we embed a real-time scheduling mechanism in CORBA to meet temporal constraints. Then, a fault-tolerant service is built into CORBA.

According to fail-stop model, our fault-tolerant services could meet the requirements of fault detecting and fault recovery. With respect to our scheduling service, the suitability of assigned priority value still needs further verification.

Keywords: Broadband Network, Fault-Tolerance, Real-Time, Real-Time Data Acquisition, Signal Processing,

Distributed Computing Environment, CORBA

一. 導論

近年來，由於通訊技術的進步和網路使用的普及，使得各項以網路系統為基礎的研究計畫和商業應用相繼展開，但在現今網路傳輸協定和網路硬體架構只能提供盡力達成(best effort)服務的限制之下，使得要以現有網路為平台，開發具有傳輸流量大及溫和即時(Soft Real-Time)要求的多媒體系統或進行需要嚴格即時(Hard Real-Time)保證的即時資料擷取與信號處理的相關研究與應用，需花費額外的心力來處理即時保證和頻寬利用的問題。

為了解決頻寬不足的問題並滿足即時性保證的要求，下一代的網路硬體架構和網路傳輸協定都針對這兩個需求提出了解決方案，本文即是以下一代的寬頻網路為基礎來進行即時資料擷取與信號處理的研究，並建立一個方便從事研究和易於發展相關應用的分散式計算環境，這個分散式計算環境的主要任務即是將底層如：寬頻網路、即時通信傳輸協定，對於即時資料處理的支援以及頻寬資源管理的機制，對應到可供程式設計者使用以開發具有即時性程式的應用程式發展界面(API)。

在建立此一分散式環境的實作上，我們採用通用物件請求仲介架構(Common Object Request Broker Architecture, CORBA)，以下簡稱CORBA，作為發展平台[OMG98]。由於 CORBA 制定了在各異質性平台上共通的溝通機制，並以物件導向的方式將這些服務機制包裝起來，加上功能完整的界面描述語言(Interface Definition Language, IDL)；使得用戶端(Client)程式只需依照伺服端(Server)所提供的界面來發出服務請求，即可透過 CORBA 的內部機制將此請求傳遞至有提供此項服務的伺服程式；即使該伺服程式是在不同的機器上執行。藉由 CORBA 提供的機制，對架構其上的應用程式而言，在不同平台間進行資料交換及程式呼叫的動作被簡化成一般的函式呼叫，所以相當適合應用於跨越多個不同平台的分散式計算上。

然而在 CORBA 目前的規格中[OMG98]，並未包含對即時性需求的支援，所以在服務請求的傳遞、服務請求的排程、緊急訊息的發送…等方面，都會有無法達到即時要求的情況發生[PPW97]。因此，我們需在現今的 CORBA 架構中加入支援即時性要求的機制，以便於在 CORBA 上建立即時資料擷取與信號處理的分散式計算環境。

除此之外，在分散式計算的領域中，容錯服務的提供[HuKi93]，一直是一個系統能否永續且正常運作的決定因素之一，但在 CORBA 的規格中，並未對在 CORBA 架構上如何提供容錯機制做出規範，為了讓我們所建立的分散式計算環境能提供穩定且持續的運算服務，我們也決定研究在 OCRBA 上提供容錯服務的方法。

在本文中，我們將提出如何在 CORBA 架構上加入可滿足即時性要求的服務機制，並說明在 CORBA 上提供容錯服務的方法，本文會先對目前所提出的相關研究成果進行討論與說明，最後再提出我們的做法。

二. 達成即時性要求的機制

雖然 CORBA 提供了解決在分散式系統中搜尋物件所在位置的方法，但在[SLM97]和[SLC98]中則指出：由於缺乏規範服務品質(Quality of Service, QoS)的界面和強制達成所要求之服務品質的機制，同時又缺乏時間性質以及未對架構進行效能的最佳化處理，將使得 CORBA 無法滿足具有即時性的服務請求。然而在要求即時性的應用越來越多的情況之下，能滿足即時性要求的 CORBA 也就成為目前在分散式物件計算 (Distributed Object Computing, DOC) 上研究發展的重點。

二.一. 對即時性質支援待改進之處

所謂的能滿足即時性要求的 CORBA(Real-Time CORBA, RT CORBA)

是指能夠對一個端到端(end-to-end)的執行程序，提供規範時間性限制的描述方式，並具有能達成該項時間性限制的機制[WoJo97]，但現今的 CORBA 標準則因為下列四項缺失，使得 CORBA 對於即時性質的支援相當地不足。

- ***規範時間性要求的方法***

目前的 CORBA 標準中並未提供規範時間限制的方法，所以有時間性要求的應用程式無法以一個共通的方式來表達所需的時間規範。除此之外，目前的 CORBA 規格中並沒有制定優先權(priority)的觀念，因此應用程式彼此間無法以優先權值的高低來分辨誰的時間限制較為緊迫，也就是說時間限制的性質在目前的 CORBA 規範中是被忽略的。

- ***非阻斷式(non-blocked)程序處理的支援***

在現今的 CORBA 規範中，一個具有回應的服務請求是以阻斷式的程序處理方式來達成的，也就是說客戶端的程式在發出服務請求後，是以忙碌等待(busy wait)的方式來獲得回應的資料，而不使用在即時系統廣泛採用的非同步(asynchronous)機制[Gal95]；就是應用程式在進入等待狀態後會教出執行權，而由系統在回應資料到達時將控制權交還原應用程式來接收回應的資料。

- ***可達成時間規範的強制機制***

在服務請求的處理策略上，CORBA 標準中並沒有針對提供即時保證而予以規範，所以目前的 CORBA 實作品大多採取“先到先執行”的方式來處理用戶端的服務請求，而不是採用優先權導向的服務處理策略，所以時間限制的急迫性無法落實在執行順序上。

另外在系統資源的管理方面，目前的 CORBA 規格中依然是付之闕如，因此任意的應用程式皆可無限制地要求系統資源，而造成時間急迫性高的任務會因資源不足而錯失達成任務的時機。

- ***針對即時保證設計的實作架構***

由於 CORBA 在制定時並未考量對於即時性質的支援，因此現今的 CORBA 實作品在設計時，並沒有針對即時系統的特性加以考量，而是以增加系統對於服務請求的處理量以及縮短回應的延遲時間為設計重點，因而使得整個系統的端對端可預測性降低。除此之外，由於目前的 CORBA 實作品內部大都採用“先進先出”佇列的架構，再加上並未提供搶奪機制(preemption)，所以一但發生優先權倒置(priority inversion)的情形，現今的 CORBA 實作品並無法提供有效的解決方法，形成要在 CORBA 系統上提供即時保證的阻礙。

二.二. 增進即時性質的相關研究

[SLM97]提出了以達到端到端的 QoS 保證為設計目標的物件仲介架構(Object Request Broker, ORB)；該篇論文假設在 CORBA 底層的作業系統和網路都能提供資源預約和即時性的排程，再加上 ORB 系統內部定義 QoS 保證的方法和滿足所需 QoS 保證的機制，藉此來達到端到端的 QoS 保證，實作時則朝向找出建構 RT CORBA 系統較有效能的架構，因此採用了許多有助於建構分散式系統的軟體設計典範(design pattern)，以實驗性的方式來對整個系統的架構和效能做最佳化，在描述 QoS 保證部份，該系統在原有的 IDL 中加入制定 QoS 保證的語法，讓程式設計者能提供所需 QoS 保證的資訊，ORB 內部再根據這些資料來進行排程，至於在達成 QoS 保證方面，採用靜態型的即時排程法則，所以在架構上有一個離線排程性的分析機制，按照服務請求的發生頻率給定優先權，再按照優先權高低來執行服務。

至於在[WoJo97]中，則提出在 CORBA

中支援時間性的分散式函式呼叫(Timed Distributed Method Invocations, TDMIs)方法來達到即時服務的要求，所謂的 TDMI 是在用戶端對伺服端發出遠端程序呼叫(remote procedure call, RPC)中，直接包含與時間相關的資訊[WBTK95]，論文中定義了用戶端的時間性要求以及伺服端的時間性規格兩種；用戶端的時間性要求主要規範了所發出 RPC 的截止時間、最早的開始時間、最近的開始時間和週期以及最多執行時間，至於伺服端的時間性規格是指該 RPC 在伺服端預期的執行時間；包含最小、平均和最多所需的執行時間。在 CORBA 中是透過 IDL 來支援這些時間性質的描述，[WoJo97] 中實作了 _after, _before, _by, _execute 四種欄位來規範用戶端呼叫伺服端所提供函式的時間特性，為了達成函式呼叫的時間限制，在 CORBA 中必須提供時間同步的機制，於是採用網路時間協定(Network Time Protocol, NTP)來達成時間的同步，另外，在各伺服端也須按照函式呼叫的重要性和時間急迫程度來排定執行的優先權，實作的方法是在各伺服端執行一個常駐程式，負責將各函式呼叫的優先權對應至所在作業系統最接近的優先權值。

而為了因應分散式物件計算對於即時性支援的要求，OMG 亦成立了 RTSIG 這個小組來議定 CORBA 對於即時性延伸上的標準規格，目前公佈的提案只針對提供端到端的可預測性來訂定標準，內容訂定了可被排程的個體型別和可被資源管理的資源型別，以及應用上述型別來滿足即時性要求的機制如：優先權的傳遞、防止優先權倒置、資源管理…等，文中所提的 RT CORBA 架構包含有即時優先權、執行緒排程、優先權的對應和執行緒資源…等新的資料型別，以及新加入的服務模組如：即時 CORBA 管理者、即時排程服務…等，除了功能描述外，也訂定了完整的界面，但在即時排程服務方面，只提出了排程服務的界面和語意上的解釋，並未決定所要使用的排程法則[RTSIG98]。

二.三. 增進即時性質機制的設計

為了加強 CORBA 對於即時保證的支援，我們採取在 CORBA 系統上建立排程服務(scheduling service)的方式，來提供規範時間特性的功能以及指派優先權值的服務。

在規範時間性質的功能上，我們定義了一個資料結構來讓應用程式設定所具有的時間特性，如下所示：

```
struct time_value {
    unsigned long tv_secs;
    unsigned long tv_usecs;
};

struct schedule_info {
    time_value exec_time;
    time_value period;
    unsigned short importance;
};
```

其中 exec_time 欄位代表任務所需的執行時間，period 則是任務的發生週期，importance 值則是當有兩個任務的執行時間和發生週期都相同時，則以個別 importance 的高低來決定彼此間的優先次序。

至於在優先權的指派策略部份，我們採用單一比率(rate monotonic)演算法來計算各個服務請求合適的優權值，單一比率演算法是以各個任務發生週期的高低來作為所指派優先權值的依據；發生週期越短的任務，被指派的優先權越高，反之，發生週期越長者，獲得的優先權值就越低[LiLa73, LSST91]。然而，並不是所有任務的執行順序都適合以單一比率演算法來進行排程；必須是能滿足單一比率演算法可排程測試(schedulability test)的任務，才能

在透過單一比率演算法排程後，可以滿足該任務所具有的即時性規範。

所謂的可排程測試是指一組要共同排程的任務，他們個別的執行時間對上發生週期的比數總和必須小於運算利用上限(utilization bound)[RHB97, GeKa96]，其運算式如下所示：

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\Lambda+\frac{C_{n-1}}{T_{n-1}}+\frac{C_n}{T_n}\le U(n)$$

其中 C_i 代表各個要排程任務的執行時間，而 T_i 則代表個別的發生週期，至於 $U(n)$ 的運算式則如下所示：

$$U(n)=n(2^{\frac{1}{n}}-1)$$

二.四. 實作方法

整個排程服務的操作過程如下所示：

- ***Step1 排程測試***

在進行排程之前，必須對所有要排程的任務進行可排程性的測試，也就是將每個任務的執行時間和發生週期代入下列的運算式：

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\Lambda+\frac{C_{n-1}}{T_{n-1}}+\frac{C_n}{T_n}\le U(n)=n(2^{\frac{1}{n}}-1)$$

當運算式成立時才進行後續的步驟。

- ***Step2 指派優先權值***

依據單一比率演算法的規定，指派較高的優先權值給發生週期較高的任務，發生週期較低的任務則獲得較低的優先權值。

- ***Step3 調整優先權值***

在指派完優先權值後，會對每個任務進行檢視，若發現具有相同優先權的任務，則再根據所具有的重要性，來對彼此的優先權進行調整。

我們實作的環境是在一台以 Red Hat Linux 5.2 版為作業系統的 Pentium-120 機器上，所使用的 C++編譯器是 egcs-1.1.2，採用的 CORBA 實作品是 MICO 2.2.7[RoPu99]，圖一中所示是排程程式的效能，縱軸是每排程一次平均所耗費的時間，橫軸則是每一次排程的物件總數。

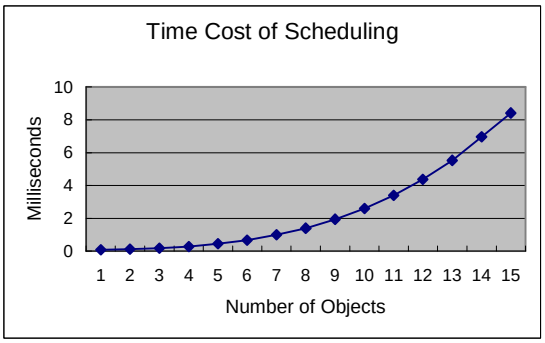

圖 一 排程所需時間圖

三. 提供容錯服務的方法

容錯機制的主要目的在於讓伺服端程式能提供連續且正確的服務，這種特性在分散式計算環境上相當重要；因為在分散式系統中，應用程式會透過遠端伺服程式來獲得運算服務或取得資料，更甚者，伺服程式本身在執行或處理服務請求時，也會應用到其他伺服器所提供的服務，而這種情況在伺服程式的設計階段式是無法得知的。在這種無法預知的相互依賴關係之下，如果沒有容錯機制來維持伺服程式的運作，那麼很有可能會因某個伺服程式的異常終止，而造成一連串其他伺服程式的不正確運作或非預期地終止服務，進而使得用戶端的應用程式停擺或發生錯誤，如此將造成難以估計的損失。

三.一. 支援容錯服務的不足之處

容錯服務依照所要處理的錯誤來源可分為硬體階層和軟體階層兩種，而軟體階層的部份又可再區分為在作業系統層和在應用程式層的容錯服務，我們在此將只討論應用程式層的容錯服務，從使用者的觀點來看，也就是指維持應用程式所提供服

務的可獲得性和所提供資料的一致性，而要達到這兩個目標，在應用程式層面必須有偵測錯誤產生和重新回復至錯誤產生前狀態的機制，由於並不考慮位在硬體層面和作業系統層上容錯機制的影響，所以我們所說的軟體容錯機制可定義為一組執行在電腦系統應用程式層面，用來偵測未被硬體和作業系統層面容錯機制處理的錯誤，並且進行回復動作的軟體元件[HuKi93]。

因此我們可採用類似共同物件服務(common object service)[OMG98-2]的方式來提供容錯機制，也就是說所需要的是一個共通的規格或要求來制定所需提供的容錯服務功能以及開放出來的界面形式，然而在目前的 CORBA 標準中並未提到任何與容錯機制有關的規範，所以在 CORBA 系統上執行的應用程式目前不僅沒有共同的方法和程序來請求容錯機制的支援，而且也不一定有容錯服務可供使用[CHY98]。這將使得以 CORBA 為基礎所發展的應用程式無法保證其所提供服務的可依賴性(reliability)。

三.二. 提供容錯機制的相關研究

[CHY98]中實作了一套提供容錯機制的服務，他們以 CORBA 的 IDL 定義出容錯服務的界面，提供 ReplicaManager 和 WatchDog 兩組界面，透過 CORBA 提供的溝通和找尋物件的機制，用戶端程式即可藉由這兩組界面來跟伺服程式要求所需的容錯服務，其中 WatchDog 負責提供錯誤偵測的服務；WatchDog 的伺服程式會週期性地發送訊息給被監控的物件並觀察物件所作的回應，或者由物件本身定期回傳訊息給 WatchDog，經由這兩種方式來確認物件的正常運作，至於 ReplicaManager 則負責提供重新起始程式的服務，當某個伺服物件因發生錯誤而崩潰時，ReplicaManager 伺服程式須起始另一個伺服物件來接手原本的服務。由於容錯服務是以伺服物件的形態出現，所以程式發展者可以自行決定容錯服務的使用方式。

[Maf95]提出的 Electra 架構則將物件群組(object-groups)和多址傳播(multi-cast)這兩項機制加入 CORBA 中，所謂的物件群組是指將多個具有同樣功能的伺服物件看作一個獨立的個體，用戶端程式直接對此個體要求服務，至於是由群組中的哪一個物件來處理，用戶端則無從得知，藉由物件群組的概念可以達到容錯服務中的主動式、被動式的伺服端複製(active, passive replication)和多重版本(multi-version)等機制，可達到連續且不中斷的容錯服務層級，而多址傳播則可應用在對群組個體的訊息傳送，為了提供多址傳播的功能，Electra 需要額外的通信傳輸層如：Horus, Isis，的支援，所以目前 Electra 是架構在 Horus 或 Isis 具有群組訊息傳遞功能的系統之上。

三.三. 容錯服務的設計

在容錯服務方面，我們希望能達到第二個層級，也就是提供錯誤偵測和重新起始程式的機制，再加上我們希望可以讓使用者自行決定所需要的容錯服務，所以我們將在 CORBA 架構中設計提供容錯服務的界面，使程式發展者可以透過這些開放的界面來發出容錯的服務請求：

- *錯誤偵測（Fault Detection）*

  專門負責監測程式的運作正常與否，需要錯誤偵測服務的程式只需向此服務界面進行註冊(register)的動作，監測程式就會按照所要求的週期來進行程式運作的偵測，當偵測到程式運作不正常之後，會發出訊息給錯誤回復程式以執行後續的處理。

- *錯誤回復（Fault Recovery）*

  負責容錯處理的策略，在某個程式發生錯誤之後，錯誤回復程式會根據該程式在註冊時所提供的資訊來決定所要實行的步驟，如：重新執行一個新的程式實體(instance)或將新的程式實體回復到發生錯誤前最後的狀態。

三.四. 實作方式

a ***錯誤偵測***

當一般的應用程式需要偵測錯誤服務的支援時，該應用程式須先向提供偵測錯誤服務的伺服程式進行註冊，同時設定每次回應的週期，如圖二中的 Subscribing 步驟。在註冊程序完成後，要求服務的應用程式須按照所設定的週期定時向伺服程式以呼叫回應函式—notifying 的方式，來告知伺服程式目標程式仍運作正常無誤，即圖六中的 Notifying 步驟。一旦有註冊的程式在超過設定的週期而沒有回應，則伺服程式會將該程式當作因發生錯誤而停止執行，而將該程式自註冊表中刪除，如圖二中之 Updating 步驟。

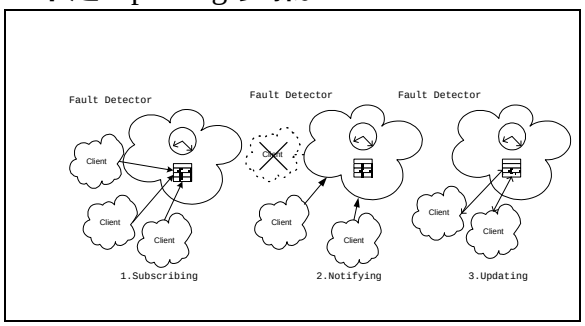

圖 二 錯誤偵測流程圖

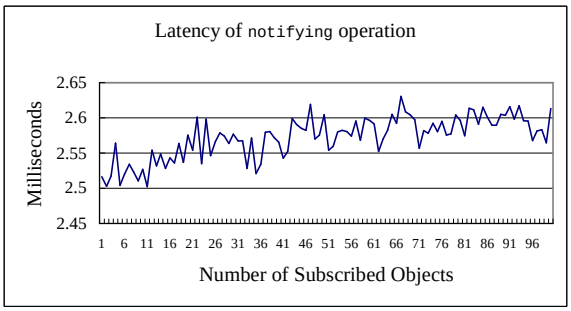

圖 三 回應函式的時間延遲圖

我們在此列出每呼叫一次回應函式平均所花費的時間與總共註冊的應用程式物件數目的對照圖：

由圖三中可以得知；即使所註冊的應用程式物件數目增加到 100 個，每呼叫一次回應函式平均所需時間大致上落在 2.5 ms 至 2.65 ms 之間，至於在 MICO 這個 CORBA 標準的實作系統上，一個既無參數亦無回傳值的函式呼叫平均需花費 1.746 ms 的時間。

a ***錯誤回復***

首先，需要重新起始程式服務的用戶程式必須向提供服務的伺服程式註冊，此時，該伺服程式會自動替該用戶程式向可獲得之提供偵測錯誤服務的伺服程式註冊要求提供偵測錯誤發生的服務，即圖四中之 Subscribe 步驟。之後，該用戶程式須在固定週期內向偵測錯誤伺服程式以呼叫回應函式的方式來回報執行狀況的正常與否，如圖中之 Notifying 步驟。當有註冊過的用戶程式超過預定時間而未回報，則偵測錯誤伺服程式會將該用戶程式視作因執行發生錯誤而停止，因而發出訊息給提供重新起始服務的伺服程式，由提供重新起始服務的伺服程式來產生新的處理程序以執行因發生錯誤而停止的用戶程式，即圖四中之 Restart 步驟。

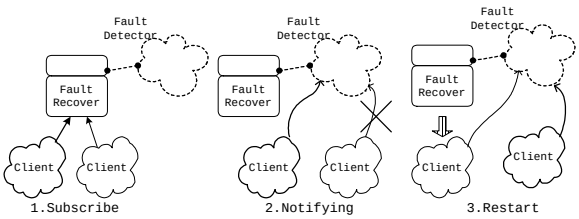

圖 四 錯誤回復操作程序

四. 結果與討論

在分散式物件計算環境中，CORBA 可說是一個廣被接受的標準，程式發展者可以自由地應用 CORBA 提供的各種服務的界面，以物件導向的概念輕易地解決主從式架構程式中最繁雜的透過網路通訊溝通的部份，現今的 CORBA 標準之中雖然定義了解決繁雜溝通程序的問題，但隨者多媒體應用的盛行和寬頻網路的發展，程式設計師開始思考在 CORBA 上架構處理具有即時性程式的可能性。

在本論文中，我們透過在 CORBA 上提供排程服務的方式，讓應用程式的執行優先次序可以藉由個別的時間限制來決定，達成了支援即時性質的第一步驟。除此之外，我們也設計了在 CORBA 中提供容錯服務的機制，分別為偵測錯誤發生以及重新起始程式服務，程式發展者可以利用偵測錯誤發生服務來檢測目標程式的運作是否正常，一旦發現目標程式因發生錯誤而停止執行，則可以透過重新起始程式服務來重新執行該程式，以達到維持服務可獲得性的要求。

經由這些對傳統的 CORBA 架構所作的延伸與增強，程式設計師可以方便地應用 CORBA 所提供的即時性質支援和容錯服務服務機制，使得所發展的程式可以達到即時性和容錯的要求。

參考文獻

[CHY98] P. Emerald Chung, Yennun Huang, Shalini Yajnik, "DOORS: Providing Fault Tolerance for CORBA Applications,"

[GeKa96] Robert George, Yutaka Kanayama, "A Rate-Monotonic Scheduler for the Real-Time Control of Autonomous Robots", Proceeding of the 1996 IEEE International Conference on Robotics and Automation, April 1996.

[Gom89] Hassan Gomaa, "Software Design Methods for Real-Time Systems", Technical Report, SEI-CM-22-1.0, Software Eng. Inst., Carnegie Mellon Univ., Dec. 1989.

[GLS98] Christopher D. Gill, David L. Levine, Douglas C. Schmidt, "The Design and Performance of a Real-Time CORBA Scheduling Service," Aug. 1998.

[HuKi93] Yennun Huang and Chandra Kintala, "Software Fault Tolerance in the Application Layer," June 1993.

[KSTW94] P. Krupp, A. Schafer, B. Thuraisingham, and V. F. Wolfe, "On real-time extension to the common object request broker architecture," in Proceedings of the Object Oriented Programming, Systems, Languages, and Applications (OOPSLA)'94 Workshop on Experiences with the Common Object Request Broker Architecture (CORBA), Sept. 1994.

[LiLa73] C. L. Liu, J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Jounal of the ACM, vol.20. No.1, 1973.

[LSST91] John P. Lehoczky, Liu Sha, J. K. Strosnider, Hide Tokuda, "Fixed Priority Scheduling Theory for Hard Real-Time Systems", *Foundations of Real-time Computing*, Kluwer Academic Publishers, 1991.

[MoZa95] Thomas J. Mowbray, Ron Zahavi, The Essential CORBA: Systems Integration Using Distributed Objects, John Wiley & Sons, Inc., New York, 1995.

[Maf95] Silvano Maffeis, "Adding Group Communication and Fault-Tolerance to CORBA", Proc. of the USENIX Conf. on Object-Oriented Tech., June 1995.

[Maf96] Silvano Maffeis, "The Object Group Design Pattern," June 1996.

[MaSc97] Silvano Maffeis, Douglas C. Schmidt, "Constructing Reliable Distributed Communication Systems with CORBA," IEEE Communications Magazine Vol. 14, NO. 2, Feb. 1997.

[OMG98] Object Management Group, The Common Object Request Broker: Architecture and Specification, Feb. 1998.

[PPW97] A. Polze, D. Plakosh, K. C. Wallnau, "A Study in the Use of CORBA in Real-Time Settings: Model Problems for the Manufacturing Domain," Technical Report CMU/SEI-97-TR-011, Software Eng. Inst., Carnegie Mellon Univ., Dec. 1997.

[PlSt98] Frantisek Plasil, Michael Stal, "An Architectural View of Distributed Objects and Components in CORBA, Java RMI, and COM/DCOM," 1998.

[PoMa98] Andreas Polze, Miroslaw Malek, "Responsive Computing with CORBA," 1998.

[RHB97] Erik Cota-Robles, James P. Held, Thomas J. Barnes, "Schedulability Analysis for Desktop Multimedia Applications: Simple Ways to Handle General-Purpose Operating Systems and Open Environments", Proceeding of IEEE International Conference on Multimedia Computing and Systems, 1997.

[RoPu99] Kay Romer, Arno Puder, *MICO is CORBA: CORBA 2.2 Impelementation*, June 1999.

[RTSIG98] The Real-Time Platform Special Interest Group of the OMG, Real-Time CORBA Joint Revised Submission, Dec. 1998.

[ShSa93] L. Sha and S. Sathaye， "Distributed Real-Time System Design: Theoretical Comcepts and Applications," Technical Report CMU/SEI-93-TR2, Software Eng. Inst., Carnegie Mellon Univ., Mar. 1993.

[SLM97] Douglas C. Schmidt, David L. Levine, Sumedh Mungee, "The Design of the TAO Real-Time Object Request Broker," Computer Communications Jounal, 1997.

[SLC98] Douglas C. Schmidt, David L. Levine, Chris

Cleeland, “Architectures and Patterns for DevelopingHign-Performance, Real-Time ORB Endsystems,” Advances in Computers, 1998.
[WoJo97] Victor Fay Wolfe, Russell Johnston, “Real-Time CORBA,” 1997.
[WBTK95] V. F. Wolfe, J. K. Black, B. Thuraisingham, and P. Krupp. “Real-Time Method Invocations in Distributed Environments”, 1995.
[YaXi97] Stephen S. Yau, Bing Xia, “An Approach to Distributed Component-based Real-Time Application Software Development,” 1997.