466434

申請日期	87. 3. -9
案號	87103433
類別	G06F 3/00

公告本

A4
C4

（以上各欄由本局填註）

發明 新型 專利說明書

一、發明新型名稱	中文	一種用在異質區域網路之物件導向式訊息傳輸界面
	英文	
二、發明創作人	姓名	1. 沈嘉雄　3. 張益郎 2. 劉育銘　4. 郭信權
	國籍	中華民國
	住、居所	1. 台北縣中和市大仁街秀水里26鄰40-1號 2. 台北縣五股鄉成泰路四段22巷10號6樓之8 3. 台北市環河北路一段99號2F 4. 台北縣汐止鎮忠孝東路459號3F
三、申請人	姓名（名稱）	財團法人資訊工業策進會
	國籍	中華民國
	住、居所（事務所）	台北市和平東路二段106號11F
	代表人姓名	果芸

裝 訂 線

經濟部中央標準局員工消費合作社印製

本紙張尺度適用中國國家標準（CNS）A4規格（210×297公釐）

五、發明說明（　）

訊息傳輸是在區域網路上各個節點之應用程式互相溝通最常用的方法，不過一般的做法是直接叫用傳輸控制通訊協定(TCP)或使用者資料封包通訊協定(User Datagram Protocal,UDP)作訊息的送與收，接收訊息的應用程式再用切換敘述(switch statement)根據訊息的種類分別去處理訊息。這種做法的缺點如下：

(1)可再用性差

直接叫用TCP或UDP的結果，使得開發新的區域網路上的應用程式時，又得把這一段程式重寫一遍。

(2)不易維護

用switch statement根據訊息的種類分別去處理訊息，當訊息的種類一多，程式變得龐大不易讀，更嚴重的是，增加新的訊息種類便得修改此段程式以及宣告訊息種類的前置檔(header file)，增加維護的困難。

(3)缺乏彈性

當區域網路的配置需要更動，例如節點由兩個變成三個或更改節點的網際網路通訊協定(ip)位址等，就得更改上述做法的程式碼；另外當區域網路上的應用程式夠複雜，訊息也夠多，此時想利用程序多工(multi-process)或執行緒多工(multi-thread)分別處理不同性質的訊息，就無法使用上述的做法，除非作大幅度的修改。

為改善習用之缺失，是以提出本案『用在異質區域網路之物件導向式訊息傳輸界面』，其中更包含一自動執行程序。

五、發明說明（　）

本案的目的在於提供一個再用性高、使用簡便、擴充性高之網路傳輸界面；其次，更提供一自動執行程序以提高應用系統設計之便利性。簡要說明如下：

本案為一種用在異質區域網路之物件導向式訊息傳輸界面，係於一包含若干節點的網路中，使一應用系統與一核心(kernel)進行通訊，以傳遞一訊息，其每一節點係包含：一訊息佇列管理器(MQM)，係由若干外向佇列(queue)與若干內向佇列組成，而藉由將該訊息寫至該外向佇列及將該訊息自該內向佇列讀出，以與該應用系統通訊；以及一訊息收發器(FMS)，係電連接至該訊息佇列管理器(MQM)，由若干外向執行緒(thread)與若干內向執行緒組成，該外向執行緒係對應至該外向佇列，藉由該外向執行緒自該外向佇列讀取該訊息及該內向執緒將該訊息寫至該內向佇列，以與該核心通訊。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該網路係為一區域網路。當然，其中該外向佇列之個數係等於該區域網路的節點數，該內向佇列係由該應用系統定義。該內向執行緒之個數係等於該區域網路除了本身外的節點數。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該內向執行緒之一係不經該網路，而直接藉該訊息收發器(FMS)將該訊息傳送至該節點本身。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該訊息收發器(FMS)係為一物件(Object)，負責由該外向佇列取出訊息送至該網路，以及由網路接收

五、發明說明（　）

訊息放到該內向佇列，該訊息收發器(FMS)更包含若干傳輸連接(TCPConnect)物件，每一傳輸連接物件各用該執行緒之一以透過一受訊(Socket)物件執行其任務。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該訊息佇列管理器係為一物件，用以維護該外向佇列與該內向佇列，更為該訊息提供一共用記憶體管理(SyncBuddy)物件以及一解決訊息跨程序(process)問題(Persist)物件。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該共用記憶體管理(SyncBuddy)物件係繼承自一Buddy物件。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該內向佇列及該外向佇列係為一同步佇列(SyncQueue)物件，繼承自一佇列(Queue)物件，再利用一Mutex物件與一CondVar物件以解決同步的問題。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中更包含：一根訊息(RootMsg)物件，使用者定義之訊息係繼承自該RootMsg物件，而該RootMsg物件係繼承自一沒有資料成員(data member)，只有虛擬法則(virtual method)的虛擬物件(PeudoRootMsg)；以及一系統資訊(SysInfo)物件，係電連接至該根訊息其包含一描述各個節點資訊的NodeInfo物件與一描述該應用系統各個程序(process)資訊的ProcInfo物件。

如所述之用在異質區域網路之物件導向式訊息傳輸界面，其中節點資訊物件係包含有一ＩＰ位址與一屬於該

五、發明說明（　）

節點的程序(Process)名稱；該程序(Process)資訊主要內容係包含一執行該程序(process)的指令名稱，一指令所在的路徑，與一或多個該程序(process)會使用的內向佇列名稱。

一種用於異質區域網路之物件導向式訊息傳輸界面之執行自動化程序，其包含下列步驟：輸入一系統初始檔(sys.ini)，藉文法剖析(Parser)以產生一系統定義檔(sysdefs.h)與一系統資訊檔(sysinfo.mmf)；輸入一訊息初始檔(msg.ini)，藉文法剖析及編碼(code)以產生一訊息定義檔(msgdefs.h)與一"*msg.h"檔；以及輸入該訊息定義檔(msgdefs.h)，藉編碼(code)以產生一"Message.h"檔與一"Persist.h"檔。

如所述之用在異質區域網路之物件導向式訊息傳輸界面之執行自動化程序，其中該"*msg.h"檔係代表個別訊息的前置檔(header file)。

本案得藉由下列圖式及詳細說明，俾得一更深入之了解：

圖一：典型的STREAM以及以TCP為例的STREAM。

圖二：本案擴充STREAM後之架構示意圖。

圖三：在包含三個節點的區域網路中，以節點三為例之收送示意圖。

圖四：本案之邏輯觀念圖。

圖五：本案靜態觀念結構。

圖六：本案實體觀念結構。

圖七：本案實體觀念結構之process view。

五、發明說明（　）

圖八：系統配置檔案自動化處理程序。

圖九：系統訊息檔案自動化處理程序。

以上圖式之主要構件如下：

11：訊息傳遞頭。

12：驅動端。

13：一對佇列。

14：傳輸控制通訊協定(TCP)。

15：網際網路通訊協定(IP)。

21：訊息佇列管理器(MQM)。

22：訊息收發器(FMS)。

OBx：外向佇列。

IBx：內向佇列。

OBTx：外向執行緒。

IBTx：內向執行緒。

本發明利用物件導向的特性與程式自動生產（automatic code generation）的技巧，並且將Dennis Ritchie發明之STREAM機制從核心面(kernel space)往外擴充到使用者面(user space)，才得到這種以宣告方式代替寫程式既簡單又實用的訊息傳輸機制。圖一左圖是典型的STREAM，圖一右圖是以TCP為例的STREAM架構，圖二是本發明將其擴充後的架構是意圖。圖一左圖說明一個典型的STREAM包括資訊傳遞頭11(stream head)，驅動端12(driver end)，與在它們之間的零或多個模組(module)。stream head提供與應用系統的介面，driver end直接與設備(device)溝

五、發明說明（　）

通，module用來處理中間過程的資料。每一個module包含一對佇列13--讀佇列(queue)與寫佇列，stream head與driver end也都包含一對佇列，寫佇列（write queue）將訊息往下由應用程式傳送到driver end，讀佇列（read queue）反向將訊息往上由driver end傳送到應用程式，中間的module也可以產生其它訊息往下或往上傳送。圖一右圖在stream head與driver end之間插入TCP14與IP15兩個module。其中驅動端通常為一區域網路(如乙太網路等)。

圖二在user space加一個訊息佇列管理器21(Message Queue Manager,MQM) module，訊息佇列管理器21(MQM)代替stream head提供與應用系統的介面，訊息佇列管理器21(MQM)同時管理維護兩類佇列--外向佇列與內向佇列，外向佇列相當於寫佇列，內向佇列相當於讀佇列。外向佇列的個數等於區域網路上節點的個數，應用系統定義用來接收訊息之佇列屬於內向佇列，因此內向佇列的個數會因應用系統定義之佇列的多寡而有所不同。訊息收發器22(Facility Message Service,FMS）是一個包含多個執行緒（thread）的process，在user space負責訊息的虛擬收送，也是與kernel space之stream head溝通的唯一介面。每一個外向佇列對應一個外向執行緒，負責由外向佇列取出訊息以便將訊息送出；內向執行緒負責接收其它節點傳來的訊息，接到後將訊息放到內向佇列，因此除了自己以外網路上有幾個節點就會有幾個內向執行緒(thread)。

圖三是一個包含三個節點之區域網路其中節點三的

五、發明說明（　）

訊息收送示意圖，節點三除了訊息收發器(FMS) process外，尚有應用系統的五個process SER1..USER5；訊息佇列管理器(MQM)有三個外向佇列(OB1,OB2,OB3)，其中OB1的訊息要送到節點一，OB2送到節點二，OB3送給自己，亦即OB3的訊息不經過網路直接藉FMS作IPC（Inter Process Communication），三個節點的process定義的內向佇列(IB1...IB7)有七個，其中屬於節點三的只有IB3，IB4，與IB5；對應於其它節點，FMS會產生一個外向執行緒(OBT)與一個內向執行緒(IBT)，例如對應節點一，一定有外向執行緒(OBT1)與內向執行緒(IBT1)，OBT1負責將OB1的訊息送到節點一，IBT1負責接收由節點一送來的訊息並且將訊息放到訊息指定的內向佇列，對應於節點三自己只有外向執行緒OBT3，負責對OB3的訊息作IPC。圖三的例子中，USER1透過訊息佇列管理器(MQM)將訊息送到節點一與節點二；USER3將訊息送給USER2，USER2從該訊息指定的內向佇列IB3接收訊息；USER4指定由IB4接收由節點一或節點二送來的訊息。

以下分別使用Alan Burns的四個觀點說明本案的架構：

●邏輯觀點說明設計本案的物件模型。

●靜態觀點說明本案軟體的組織結構。

●動態觀點說明設計時併行與同步的問題。

●實體觀點用分散式的觀點說明如何將軟體映射至硬體。

五、發明說明（　）

邏輯觀點之架構

本案邏輯觀點之架構如圖四所示，主要有四種類別(class)分說明如下：

FMS物件負責由外向佇列取出訊息送到網路，或由網路接收訊息放到內向佇列，當應用系統共有N個節點，針對自己以外的節點，每一個節點各需要一個傳輸連接(TCPConnect)物件從事網路接收與分送訊息的作業，因此FMS就需要用到2*(N-1)個TCPConnection 物件。每一個TCPConnection各自用一個執行緒透過受訊(Socket)物件執行它的任務，因此FMS至少要用到2*(N-1)個執行緒物件。

訊息佇列管理器(MQM)物件管理維護外向佇列與內向佇列並且是所有的process存取這些佇列的唯一窗口。當應用系統共有N個節點，訊息佇列管理器(MQM)管理N個外向佇列與所有應用系統(AP)定義的內向佇列，另外就是為訊息提供共用記憶體管理(SyncBuddy)物件以及用來解決訊息跨process問題的Persist物件。訊息佇列管理器(MQM)管理的外向佇列與內向佇列都是同步佇列物件SyncQueue，同步佇列物件繼承自Buddy物件，再利用Mutex物件解決同步的問題。

根訊息(RootMsg)物件是訊息的源頭，所有訊息都直接間接繼承自RootMsg物件。為了解決不同的編譯器(compiler)可能對同一個訊息物件在記憶體會有不同的配置(layout)，RootMsg繼承自一個沒有資料成員(data member)只有虛擬法則(virtual method)的虛

五、發明說明（　）

擬物件PeudoRootMsg。

系統資訊(SysInfo)物件根據系統配置檔案描述整個系統的配置，包含描述各個節點資訊的NodeInfo物件與描述各個process資訊的ProcInfo物件，當系統有群組時，尚包含描述各個群組資訊的GroupInfo物件。群組資訊描述屬於該群組的節點名稱；節點資訊主要的內容有IP的位址與屬於該節點的process名稱；process 資訊主要的內容有執行該process 的指令名稱、指令所在的路徑，與該process會使用的內向佇列名稱。SysInfo物件是外界詢問系統配置資訊的唯一窗口。

靜態觀點之架構

本案分成四個層次，每一層所屬的物件如圖五與示。最底層是直接使用作業系統library的作業系統層，目前使用的作業系統為NT與Solaris，硬體可以使用PC或SUN工作站。

第二層是與作業系統無關的基本物件，有些物件例如 Queue、 Buddy、 SysInfo、 Persist等使用的library 是ANSI-C，它們的code在NT與Solaris 是一樣的。但是像Thread、Mutex、CondVar、Shm等，就必須藉由bridge、abstract factory、與singleton三種design pattern，將它們作成與作業系統無關的物件，如此可將到處散置的#ifdef NT、#ifdef SOLARIS全部縮減集中到abstract factory內。

第三層是應用系統介面層次，本層次的物件是由基本物件組合而成，以便提供應用系統最簡易的使用介面。

五、發明說明（　）

例如TCPConnection利用Socket組合而成；訊息佇列管理器（MQM）由SyncQueue與SyncBuddy組合而成，而SyncQueue由Queue、Mutex與CondVar組合而成，SyncBuddy由Buddy 與Mutex 組合而成。

第四層是應用系統層次，此層只提供FMS物件，它透過TCPConnection 與訊息佇列管理器（MQM）提供的service ，以及利用Thread讓訊息可以在網路上各個節點之間流通。而其它process定義的物件全部屬於這一層，他們只要透過訊息佇列管理器（MQM）提供的service，便可以完成訊息的收送以及收到訊息後的處理。

動態觀點之架構

由於SyncQueue與SyncBuddy是同一個節點各個process共用的資源，下列幾種情形會產生併行控制的問題：

（1）當應用系統（AP）將訊息放到外向佇列的同時，對應此外向佇列之FMS執行緒也可能要從外向佇列拿出訊息。

（2）當一個節點有不只一個AP，而且這幾個AP正好同時要將訊息送往一個節點時，亦即搶著將訊息放到該節點對應的外向佇列。

（3）第一個情況AP在跟SyncBuddy要記憶體的同時，可能FMS執行緒要將記憶體還給SyncBuddy。

（4）第二個情況同時有幾個AP跟SyncBuddy 要記憶體。

五、發明說明（　）

(5)當AP要從內向佇列拿出訊息的同時，對應此內向佇列之FMS執行緒也可能要將訊息放到內向佇列。

(6)當一個節點有不只一個AP，而且這幾個AP正好同時要從同一個內向佇列拿出訊息。

(7)第五個情況AP要將記憶體還給SyncBuddy的同時，可能FMS執行緒正在要跟SyncBuddy要記憶體。

(8)第六個情況同時有幾個AP有將記憶體還給SyncBuddy。

綜合上述分析，我們知道只要每一個SyncQueue與SyncBuddy各自使用自己的Mutex，在作Queue或Buddy的動作之前先利用Mutex的Lock method將相關的動作鎖住，避免其它執行緒中途插進來執行，等到動作完全結束後，再利用Mutex的Unlock method把鎖解開，讓等著執行的執行緒進來執行。

實體觀點之架構

我們用一個較簡單的終端雷達系統來說明實體觀點的架構。如圖六所示共有六個節點，分成三組，第一組是監控子系統(Monitor Control System)有兩個節點彼此互為備援；第二組是終端雷達子系統(Terminal Radar System)也有兩個節點彼此互為備援；第三組是管制雷達席(Controller Radar Position)有兩個節點，此組不必備援，因此不必恰有兩個節點不可，視虛擬需要可以有一個或一個以上的節點。

各組功能說明如下：監控子系統要知道各個節點軟硬體的狀態，因此各個節點要將自己的狀態利用監控訊息

五、發明說明（　）

跟監控系統報告，也經由監控子系統下達監控命令到其它節點；終端雷達子系統接收雷達訊號，經過追蹤處理後，要將飛行物最新的位置傳送給管制雷達席；管制雷達席提供雷達顯示幕，方便管制員管制飛行物。互為備援的兩個節點彼此要傳送備援訊息，以便知道對方的狀態，一發生問題可以即時處理。

●圖七顯示各個節點所執行的process，每一個節點都要先執行FMS process以便傳遞訊息，因為LMC(Local Monitor Control) process控制FMS以外的process完成容錯功能，每一個節點都要執行LMC。

●TSC(Time Synchronization Control) Process 用來協調統一各個節點的內部時間，所以每一個節點都要執行TSC。

●GMD(Global Monitor Control) process提供圖形使用者介面讓維護人員知道系統各個節點軟硬體的狀態，也可以下監控命令到各個節點。GMD只在監控子系統執行。

●MDF(Monitor Data Format) process是GMD對外聯絡的代表。不管是接收各節點LMC提供的狀態報告訊息或下達監控命令訊息給某節點的LMC，都藉由MDF完成。MDF收到狀態報告訊息後，將狀態紀錄在與GMD共用的記憶體，作為GMD顯示資料的來源；當有人下命令，GMD也將命令紀錄在與MDF共用的記憶體，MDF看到後將之轉成訊息送出去。MDF只在監控子系統執行。

●RDP(Radar Data Processing)process提供接收雷達訊號與雷達訊號追蹤處理等功能，追蹤的結利果利

五、發明說明（　）

用訊息傳送給管制雷達席。RDP只在終端雷達子系統執行。

●DSP(Display) process提供圖形使用者介面讓管制員可以知道飛行物的最新位置與下達管制命令。DSP只在管制雷達席執行。

●RDF(Radar Data Format) process是DSP對外聯絡的代表。RDF收到RDP傳來的雷達訊息後，將資料紀錄在與DSP共用的記憶體，作為DSP顯示資料的來源；當管制員下命令，DSP也將管制命令紀錄在與RDF共用的記憶體，RDF看到後將之轉成訊息送給RDP處理。RDF只在監控子系統執行。

圖七中實線代表節點與節點間的訊息傳輸通道，節點間的訊息透過FMS傳送到其它節點的process，節點內的訊息是LMC與同節點的其它process(FMS除外)溝通之訊息，事實上這些訊息的傳遞也是透過FMS。

I.使用程序說明

本發明的使用程序如下：

1.使用者準備輸入檔案

A.整個應用系統定義一個系統配置檔案"sys.ini"。

此檔案說明系統的節點與process，節點的說明包括ip位址與在該節點執行的所有process名稱，process的說明包括執行檔案名稱與用來接收訊息的所有內向佇列名稱，其文法見附錄一。

B.整個應用系統定義一個訊息檔案"msg.ini"。

五、發明說明（　）

此檔案說明全部的訊息，訊息的說明包括目的地、用來接收此訊息的內向佇列名稱、用來處理此訊息的物件名稱、與此訊息所攜帶資料的資料結構等，其文法見附錄二。當應用系統很複雜，其訊息種類上百或千種，此時將所有訊息宣告在同一檔案不易維護，可將訊息分別定義在多個檔案，而由訊息檔案"msg.ini"含括進來。

C.每一個節點定義自我節點檔案"self.ini"

每一個節點都要有此檔案，說明目前所在節點的名稱。

II.執行自動化程序

A. 輸入"sys.ini"經過同樣文法的Parser先後處理兩次分別產出"sysdefs.h"與"sysinfor.mmf"，見圖八。

"sysdefs.h"的內容除了定義所有節點、process、與內向佇列的ID外，尚有節點總數、內向佇列總數等常數。

"sysinfo.mmf"是SysInfo物件的binary檔案，只有它與"self.ini"是所process在run time執行要用到的兩個外部檔案。

B. 輸入"msg.ini"經過Parser與Code Generation處理後產出"msgdefs.h"與"*Msg.h"，見圖九左圖。

"msgdefs.h"的內容定義所有訊息的ID與訊息總數。

"*msg.h"代表個別訊息的header file。

五、發明說明（　）

C. 輸入"msgdefs.h"經過Code Generation處理後產出"Message.h"與"Persist.h"，見圖九右圖。

"Mssage.h"定義之Message物件，其所佔的記憶體視所有訊息中最大者，接收訊息一律使用此物件保證不會錯。

"Persist.h"用來解決訊息在不同的process會用到不同的virtual table作dynamic binding的問題。如何解決對使用者而言是透通的，他不會用到此物件。

III. 前面步驟B中自動產生的各個訊息header file，除了處理訊息的method : Proc ()未定義外，其它的method 都自動定義好了，定義Proc()建議使用XXX::Instance()->HandleYYY(this)，其中XXX是應用程式的物件類別並且只會用到一個instance，否則要使用global物件；YYY是訊息的類別。這一個statement 是唯一需要使用者寫的code。另外，說明應用程式送訊息的範例如下：

```
XyMsgData data("Hi, send a msg to you"); // prepare message data
XyMsg msg = XyMsg(data);  //get a local message that is an instance of
XyMsg
MQM::Instance()->Send(msg); // send the local message
```

應用程式收訊息的範例如下：

```
Message msg; // get a maximum sise local msg used to receive any kind of
msg.
while(1) {
```

五、發明說明（　）

```
    MQM::Instance()->Receive(MY_Q, msg); // receive message from queue:MY_Q
    msg.Proce();                         // process the received message
}
```

IV. compile-and-link應用程式

V. 執行"fms.exe"。

用最簡單的例子說明上述五個步驟，此例只包含兩個節點，節點NODE1的AP1_PROC送訊息ToAP2Msg給節點NODE2的AP2_PROC，AP2_PROC由內向佇列AP2_Q接到後，回送acknowlege訊息ToAP2AckMsg，AP1_PROC由內向佇列AP1_Q接收此訊息。步驟一之系統配置檔案"sys.ini"內容如下：

```
〔Nodes〕
  NODE1
      ip_addrs = 128.12.23.88      //允許一個以上
      processes = AP1_PROC         //允許一個以上
  NODE2
      ip_addr = 128.12.23.89
      processes = AP2_PROC
〔Processes〕
  AP1_PROC
      exec = "ap1.exe"
      queues = AP1_Q    //允許一個以上
  AP2_PROC
```

五、發明說明（　）

exec = "ap2.exe"

queues = AP2_Q

步驟一之訊息檔案"msg.ini"內容如下：

ToAP2Msg

to = NODE2

queue = AP2_Q

data = {

char msg[128];

}

ToAP2AckMsg

to = NODE1

queue = AP1_Q

步驟一之節點一的"self.ini"內容如下：

current node = NODE1

步驟一之節點二的"self.ini"內容如下：

current node = NODE2

步驟二執行一個shell script 即可。

步驟三在步驟二自動產生之header file 補上::Proc() method的內容，例如在ToAP2Msg.h中加入斜體字的部份：

ToAP2Msg::Proc(){ *Ap20bj::Instance()->HandleToAP2Msg(this);* }

在ToAP2AckMsg.h中加入斜體字的部份：

ToAP2AckMsg::Proc(){*Ap20bj::Instance()->HandleToAP2AckMsg(this);*}

接著依步驟三的範例發展應用程式，AP1_PROC有關訊息的程式如下：

五、發明說明（　）

```
class Ap1Obj(){
   ...
   Ap1Obj::Ap1Obj() {...}
public:
   static Ap1Obj * Instance() {...}
   void HandleToAP2AckMsg(ToAP2AckMsg *msg) {...}
   ...
}

main()
{
   ToAP2MsgData data("Hi, send a msg to you"); // prepare message data
   ToAP2Msg msg = ToAP2Msg(data); // construct message
   MQM::Instance()->Send(msg);  // send the local message
   Message msg;
   while(1) {
       MQM::Instance()->Receive(AP1_Q, &msg); // receive message
       msg.Proc();          // process the received message
   }
}
```

AP2_PROC有關訊息的程式如下：

```
class Ap2Obj {
   ...
   Ap2Obj::Ap2Obj(){...}
public:
```

五、發明說明（　）

```
    static Ap20bj * Instance() {...}
    void HandleToAP2Msg(ToAP2Msg *msg)    { ToAP2AckMsg msg;
                                            MQM::Instance()->Send(msg);
                                            }
    ...
}

main()
{
    Message msg;
    while(1) {
        MQM::Instance()->Receive(AP2_Q,&msg); // receive message
        msg.Proc();          //process the received message
    }
}
```

步驟五執行"fms.exe"各節點的FMS process會自動執行該節點的所有的process。這個例子雖然簡單，但是即使複雜如航管系統的使用程序與方式也是如此。

本案訊息傳輸機制的優點如下：

(1)可再用性高：

不管是簡單的分散式應用系統或複雜的航管系統都適用，內部為執行緒同步化(thread synchronization)，記憶體管理等產生的跨作業系統物件其可再用性更高。

(2)傳輸的訊息只是單純的資料而是訊息物件，因此

五、發明說明(　　)

可以運用動態結合(dynamic binding)的技術取代switch statement ，即使增加新的訊息，應用程式也無需修改，編譯(compile)，只要與新增的訊息物件的物件碼(object code)重新連結(link)即可，增加應用系統的維護性。

(3)使用簡便：

應用系統只用到收與送兩個服務法則(service method)以及訊息物件處理訊息的method,甚至連訊息物件的程式碼都是用宣告的方式自動產生。

(4)本案提供單獨的程序(process)虛擬負責訊息的傳送，應用系統可以完全不知道此process 的存在，亦即訊息的傳輸對開發者而言是透通的。並且此process不因系統配置的改變或訊息的增加需要重新compile和link。

(5)擴充性高：

由於借用STREAM的觀念，因此可依不同的需求，實作各種通訊協定。

(6)允許使用多重區域網路，節點上應用程式的發展允許使用multi-process與multi-thread。

當區域網路上各個節點使用相同的電腦，我們稱為同質區域網路；否則稱為異質區域網路。使用異質區域網路又有字元次序(byte order)的問題。目前市面上雖然有TCP 類程式庫(class library)出售，解決了第一個缺點，但是可再用性的層級仍然太低。理想的做法是應用程式只須將訊息資料準備好就可以利用物件的method傳

五、發明說明（　）

送出去，接收訊息的應用程式只須準備好處理訊息的物件，接到訊息後控制權直接交給處理訊息的物件之method，寫應用程式只管訊息的收送，至於訊息如何送、如何收以及前面提及的各種缺點與問題，完全交給本發明代為處理，對寫應用程式的人而言，這一段是透通的（transparent）。

本案得由熟悉本技藝之人士任施匠思而為諸般修飾，然皆不脫如附申請專利範圍所欲保護者。

A5
B5

四、中文發明摘要（發明之名稱：　　）

一種用在異質區域網路之物件導向式訊息傳輸界面

本案為一種用在異質區域網路之物件導向式訊息傳輸界面，係於一包含若干節點的網路中，使一應用系統與一核心(kernel)進行通訊，以傳遞一訊息，其每一節點係包含：一訊息佇列管理器(MQM)，係由若干外向佇列(queue)與若干內向佇列組成，而藉由將該訊息寫至該外向佇列及將該訊息自該內向佇列讀出，以與該應用系統通訊；以及一訊息收發器(FMS)，係電連接至該訊息佇列管理器(MQM)，由若干外向執行緒(thread)與若干內向執行緒組成，該外向執行緒係對應至該外向佇列，藉由該外向執行緒自該外向佇列讀取該訊息及該內向執緒將該訊息寫至該內向佇列，以與該核心通訊。

英文發明摘要（發明之名稱：　　）

六、申請專利範圍

1．一種用在異質區域網路之物件導向式訊息傳輸界面，係於一包含若干節點的網路中，使一應用系統與一核心(kernel)進行通訊，以傳遞一訊息，其每一節點係包含：

一訊息佇列管理器(MQM)，係由若干外向佇列(queue)與若干內向佇列組成，而藉由將該訊息寫至該外向佇列及將該訊息自該內向佇列讀出，以與該應用系統通訊；以及

一訊息收發器(FMS)，係電連接至該訊息佇列管理器(MQM)，由若干外向執行緒(thread)與若干內向執行緒組成，該外向執行緒係對應至該外向佇列，藉由該外向執行緒自該外向佇列讀取該訊息及該內向執緒將該訊息寫至該內向佇列，以與該核心通訊。

2．如申請專利範圍第1項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該網路係為一區域網路。

3．如申請專利範圍第2項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該外向佇列之個數係等於該區域網路的節點數，該內向佇列係由該應用系統定義。

4．如申請專利範圍第2項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該內向執行緒之個數係等於該區域網路除了本身外的節點數。

5．如申請專利範圍第2項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該內向執行緒之一係不經該網路，而直接藉該訊息收發器(FMS)將該訊息傳送至該節點本身。

六、申請專利範圍

6．如申請專利範圍第1項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該訊息收發器(FMS)係為一物件(Object)，負責由該外向佇列取出訊息送至該網路，以及由網路接收訊息放到該內向佇列，該訊息收發器(FMS)更包含若干傳輸連接(TCPConnect)物件，每一傳輸連接物件各用該執行緒之一以透過一受訊(Socket)物件執行其任務。

7．如申請專利範圍第1項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該訊息佇列管理器係為一物件，用以維護該外向佇列與該內向佇列，更為該訊息提供一共用記憶體管理(SyncBuddy)物件以及一解決訊息跨程序(process)問題(Persist)物件。

8．如申請專利範圍第7項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該共用記憶體管理(SyncBuddy)物件係繼承自一Buddy物件。

8．如申請專利範圍第1項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中該內向佇列及該外向佇列係為一同步佇列(SyncQueue)物件，繼承自一佇列(Queue)物件，再利用一Mutex物件與一CondVar物件以解決同步的問題。

9．如申請專利範圍第1項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中更包含：

一根訊息(RootMsg)物件，該訊息係繼承自該RootMsg物件，而該RootMsg物件係繼承自一沒有資料成員(data member)，只有虛擬法則(virtual method)

六、申請專利範圍

的虛擬物件(PeudoRootMsg)；以及

一系統資訊(SysInfo)物件，係電連接至該根訊息其包含一描述各個節點資訊的NodeInfo物件與一描述該應用系統各個程序(process)資訊的ProcInfo物件。

１０．如申請專利範圍第９項所述之用在異質區域網路之物件導向式訊息傳輸界面，其中節點資訊物件係包含有一ＩＰ位址與一屬於該節點的程序(Process)名稱；該程序(Process)資訊主要內容係包含一執行該程序(process)的指令名稱，一指令所在的路徑，與一該程序(process)會使用的內向佇列名稱。

１１．一種用於異質區域網路之物件導向式訊息傳輸界面之執行自動化程序，其包含下列步驟：

輸入一系統初始檔(sys.ini)，藉文法剖析(Parser)以產生一系統定義檔(sysdefs.h)與一系統資訊檔(sysinfo.mmf)；

輸入一訊息初始檔(msg.ini)，藉文法剖析及編碼(code)以產生一訊息定義檔(msgdefs.h)與一"*msg.h"檔；

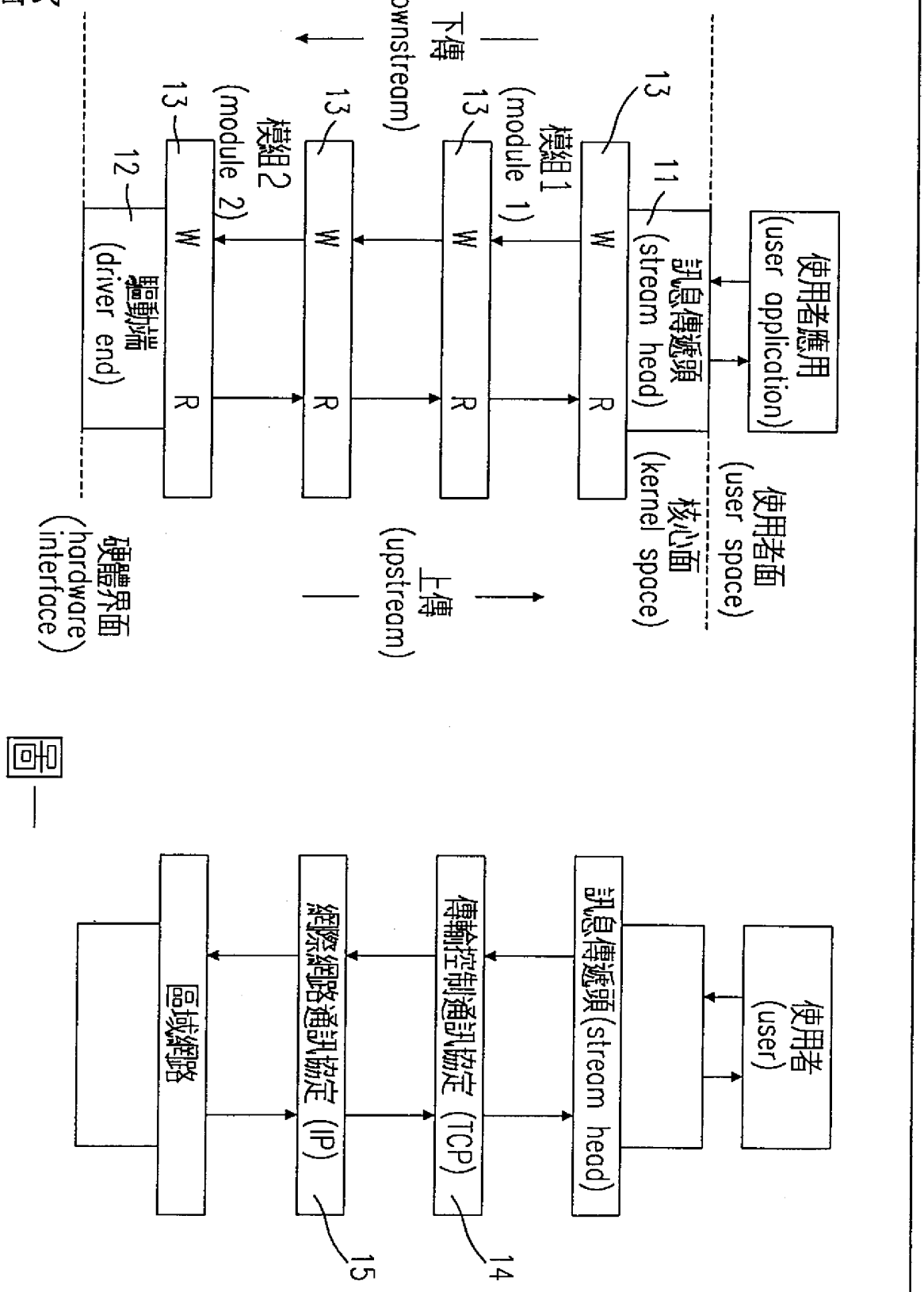
輸入該訊息定義檔(msgdefs.h)，藉編碼(code)以產生一"Message.h"檔與一"Persist.h"檔。

１２．如申請專利範圍第１１項所述之用在異質區域網路之物件導向式訊息傳輸界面之執行自動化程序，其中該"*msg.h"檔係代表個別訊息的前置檔(header file)。

A9
B9
C9
D9

87103433

圖式



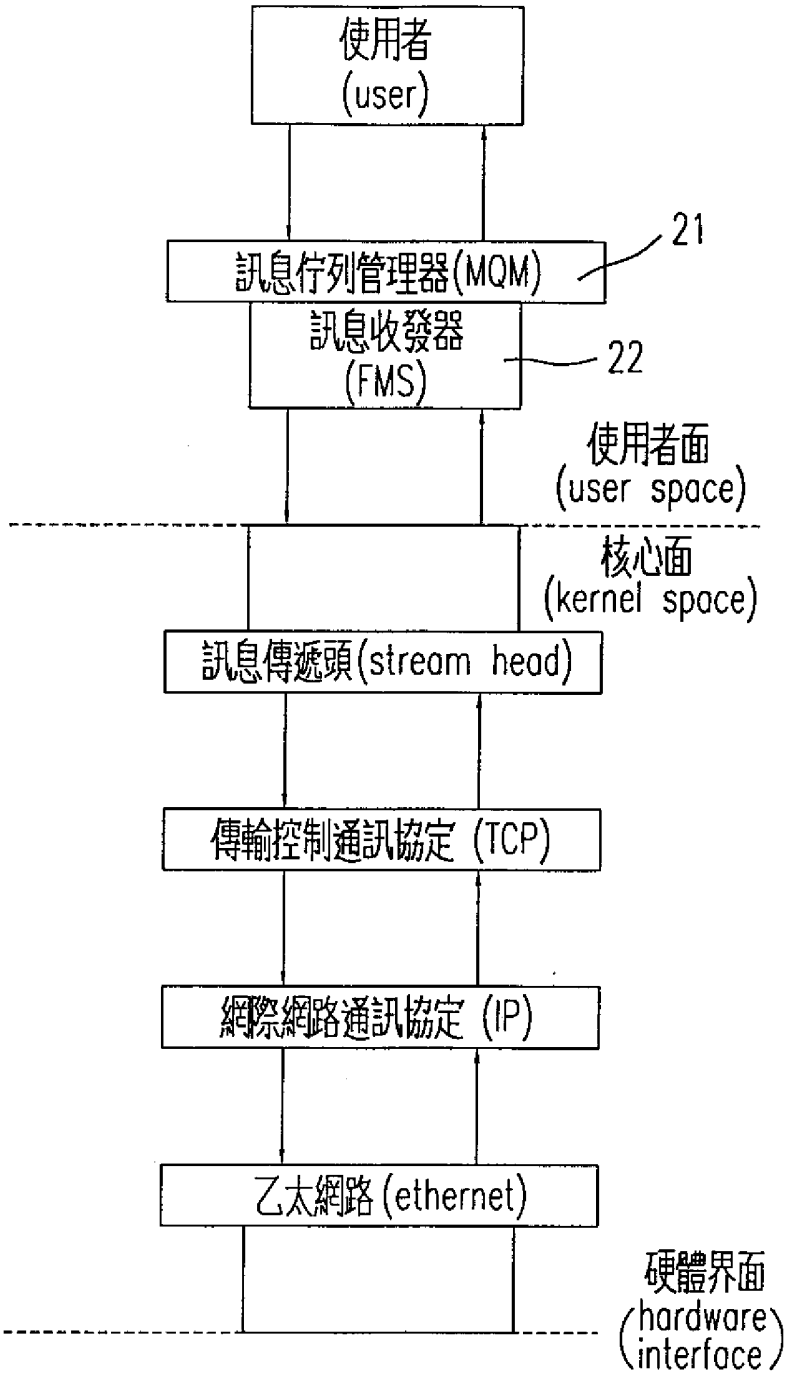
圖一

（請先閱讀背面之注意事項再行繪製）

裝 訂 線

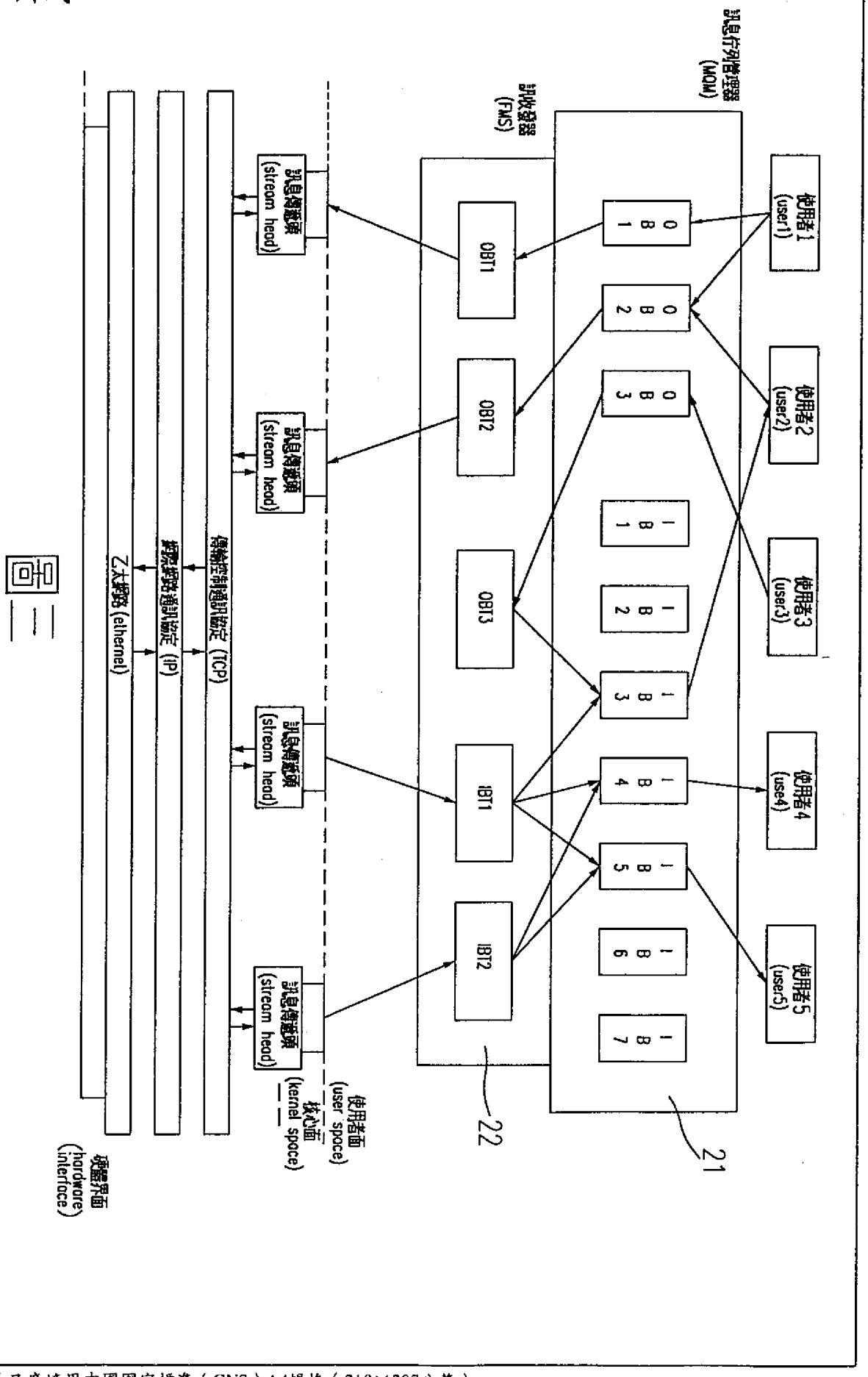
經濟部中央標準局員工消費合作社印製

圖式



圖二

圖式

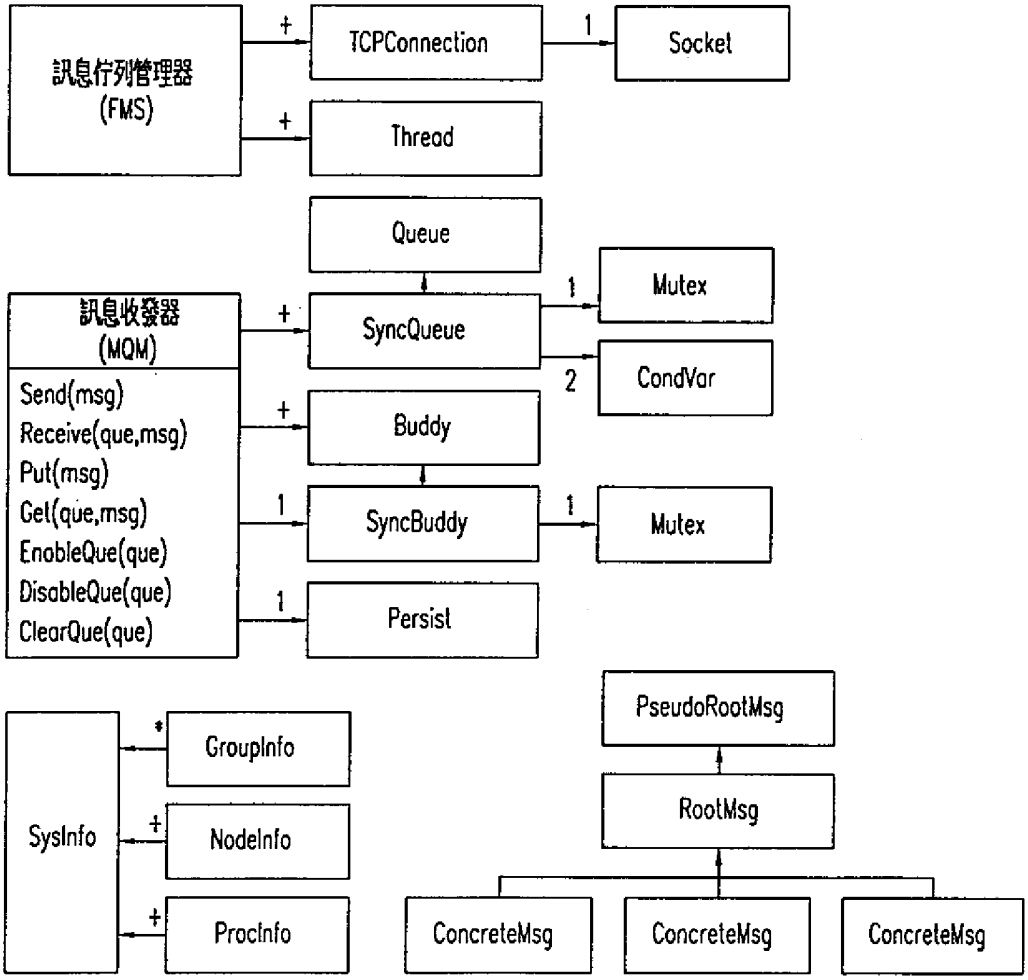


圖二

（請先閱讀背面之注意事項再行繪製）

裝 訂 線

圖式



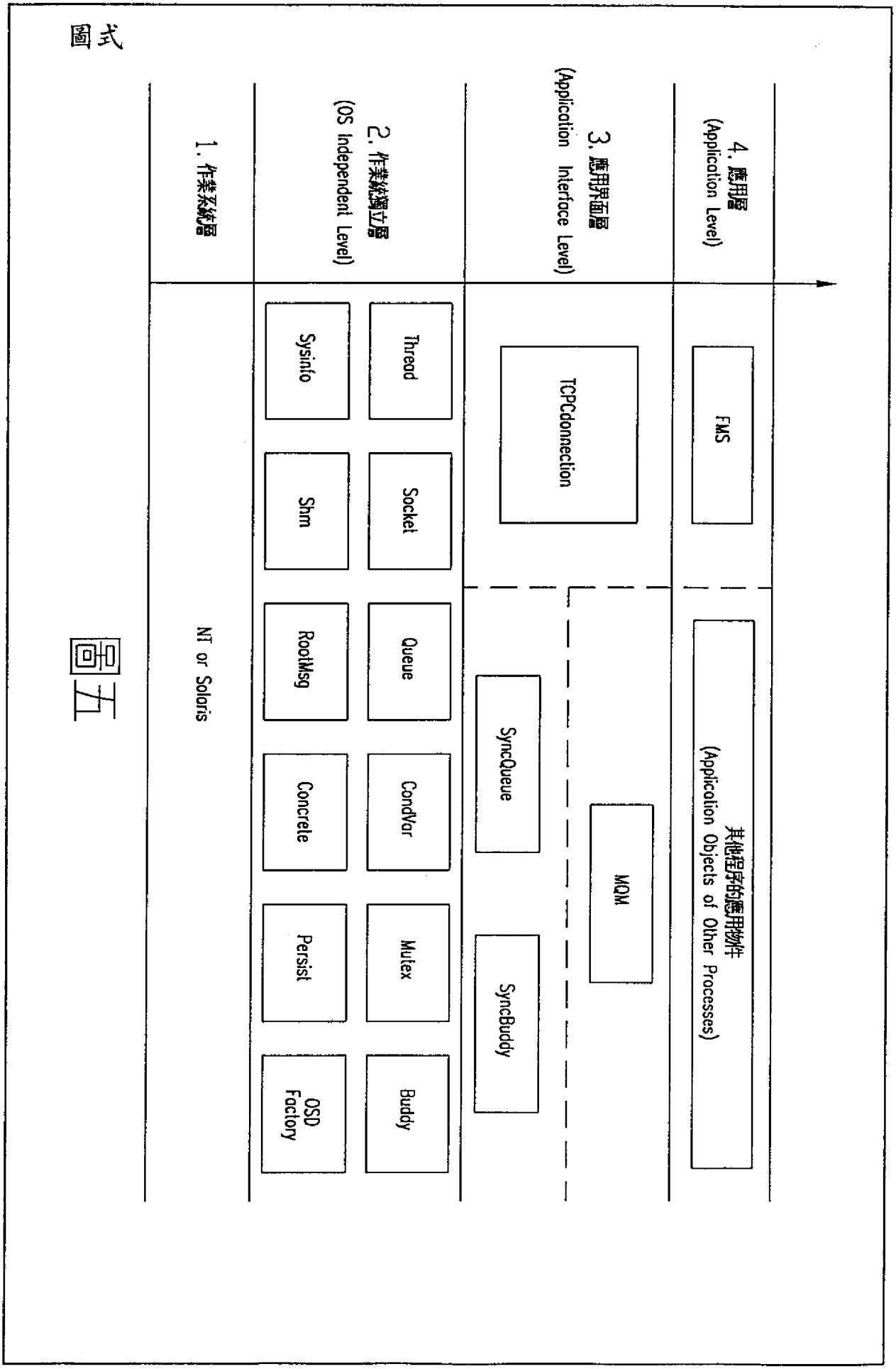
圖四

（請先閱讀背面之注意事項再行繪製）

裝 訂 線

A9
B9
C9
D9

圖式

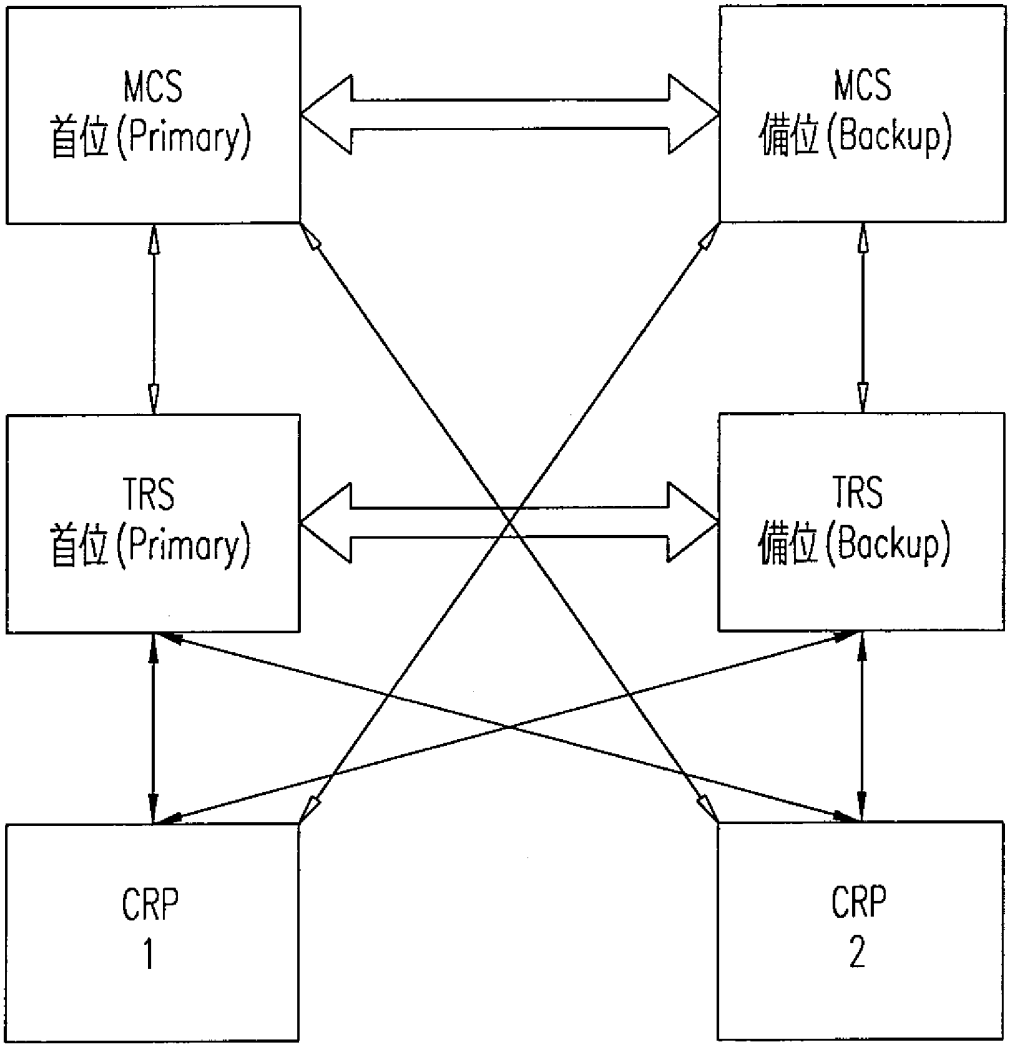


圖五

（請先閱讀背面之注意事項再行繪製）

裝 訂 線

圖式

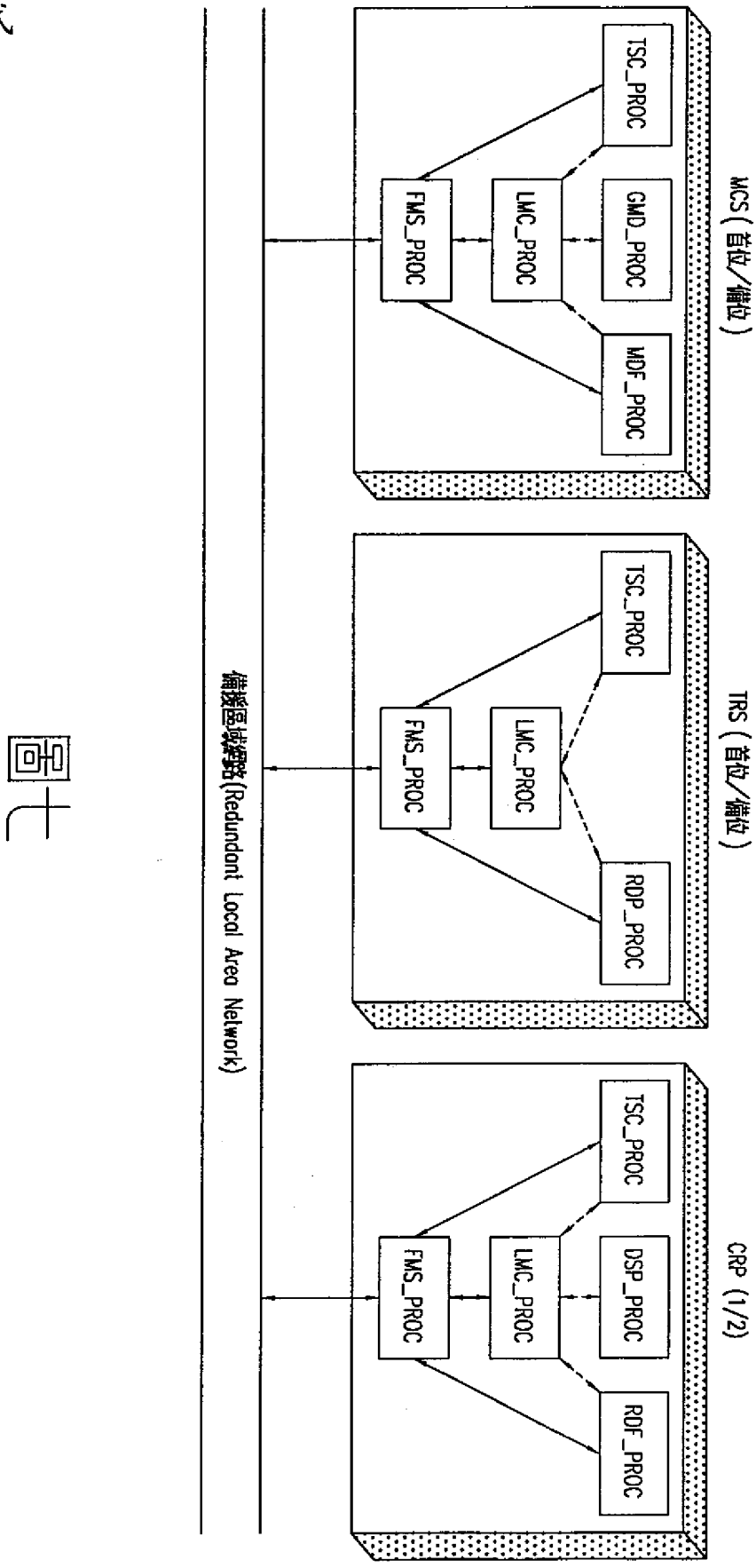


MCS：監督控制子系統
TRS：終端雷達子系統
CRP：控制雷達席

⟺ 備援管理訊息
⟷ 監督/控制訊息
⟷ 雷達顯示/命令訊息

圖六

圖式

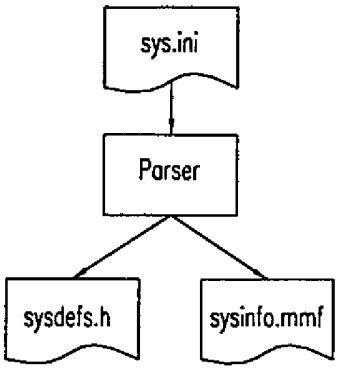


圖十

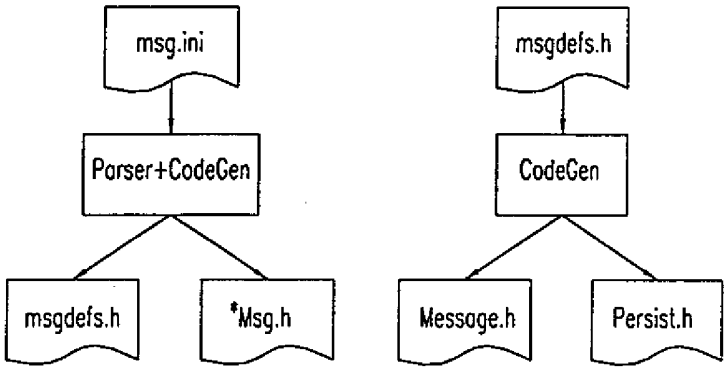
（請先閱讀背面之注意事項再行繪製）

裝 訂 線

圖式



圖八



圖九