



(19)中華民國智慧財產局

(12)發明說明書公告本 (11)證書號數：TW I477127 B

(45)公告日：中華民國 104 (2015) 年 03 月 11 日

(21)申請案號：102104061

(22)申請日：中華民國 102 (2013) 年 02 月 01 日

(51)Int. Cl. : **H04L29/06 (2006.01)****H04L12/861 (2013.01)**

(30)優先權：2012/02/03 美國

61/595,003

2012/09/15 美國

13/620,920

(71)申請人：蘋果公司 (美國) APPLE INC. (US)

美國

(72)發明人：瑪斯普特拉 凱雅 MASPUTRA, CAHYA (US)；布忽瑪 派德馬瓦西 BHOOMA, PADMAVATHY (US)；劉 喬 LIU, JOE (US)

(74)代理人：陳長文

(56)參考文獻：

TW 512602

TW I319672

TW I326544

EP 1376948A1

US 7840685B1

審查人員：蔡鴻璟

申請專利範圍項數：45 項 圖式數：17 共 82 頁

(54)名稱

用於排程封包傳輸的電腦實施方法、機器可讀媒體和客戶端裝置

COMPUTER-IMPLEMENTED METHOD, MACHINE-READABLE MEDIUM AND CLIENT DEVICE FOR SCHEDULING PACKET TRANSMISSION

(57)摘要

一種用於在一客戶端裝置上管理封包排程之電腦實施方法。舉例而言，一方法之一項實施例包含：接收待傳輸之一封包；在一網路堆疊層級處將該封包排入一佇列中；判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。

A computer-implemented method for managing packet scheduling on a client device. For example, one embodiment of a method comprises: receiving a packet to be transmitted; enqueueing the packet in a queue at a network stack level; determining whether packet scheduling is currently being performed at a driver level or at a networking stack level; selecting the packet for transmission from the queue at the network stack level if scheduling is currently being performed at the network stack level; and selecting the packet for transmission from the queue at the driver level if scheduling is currently being performed at the driver level.

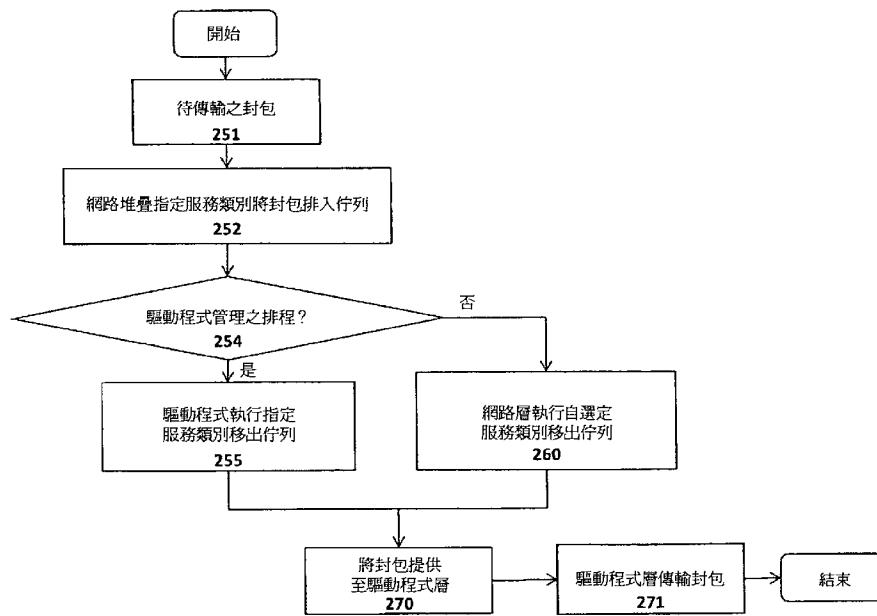


圖2B

發明摘要

公告本

※ 申請案號：102104061

※ 申請日：102年2月1日

※IPC 分類：H04L 29/06 (2006.01)
H04L 12/861 (2013.01)

【發明名稱】

用於排程封包傳輸的電腦實施方法、機器可讀媒體和客戶端裝置
 COMPUTER-IMPLEMENTED METHOD, MACHINE-READABLE
 MEDIUM AND CLIENT DEVICE FOR SCHEDULING PACKET
 TRANSMISSION

【中文】

一種用於在一客戶端裝置上管理封包排程之電腦實施方法。舉例而言，一方法之一項實施例包含：接收待傳輸之一封包；在一網路堆疊層級處將該封包排入一佇列中；判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。

【英文】

A computer-implemented method for managing packet scheduling on a client device. For example, one embodiment of a method comprises: receiving a packet to be transmitted; enqueueing the packet in a queue at a network stack level; determining whether packet scheduling is currently being performed at a driver level or at a networking stack level; selecting the packet for transmission from the queue at the network stack level if scheduling is currently being performed at the network stack level; and selecting the packet for transmission from the queue at the driver level if scheduling is currently being performed at the driver level.

I477127

103.12.23
年月日修正替換頁

【代表圖】

【本案指定代表圖】：第（2B）圖。

【本代表圖之符號簡單說明】：

(無元件符號說明)

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

(無)

發明專利說明書

(本說明書格式、順序，請勿任意更動)

【發明名稱】

用於排程封包傳輸的電腦實施方法、機器可讀媒體和客戶端裝置
IMPLEMENTED METHOD, MACHINE-READABLE MEDIUM
AND CLIENT DEVICE FOR SCHEDULING PACKET
TRANSMISSION

對優先權之主張

● 本申請案係關於 Cahya Masputra 等人在 2012 年 2 月 3 日申請之題為「SYSTEM AND METHOD FOR INTELLIGENT NETWORK QUEUE MANAGEMENT」的美國臨時專利申請案第 61/595,003 號，且主張該案之權利，該案之全文據此以引用之方式併入本文中。

對相關申請案之交叉參考

本申請案係關於讓渡給 Apple 之同在申請中的美國申請案第 13/620,920 號，該案係由 Cahya Masputra 等人在 2012 年 9 月 15 日申請。

【技術領域】

● 本發明之實施例係有關在客戶端裝置中管理資料網路通信。亦描述其他實施例。

【先前技術】

資料網路允許人們使用其「連在網路上」之各別客戶端裝置來彼此通信，並自網路上之各種源獲得資訊。舉例而言，在使用者之工作站或膝上型電腦中執行之 Web 瀏覽器應用程式可與 Web 伺服器連接，以下載 Web 網頁。連接可橫越網路之若干中間節點或躍點，中間節點或躍點可包括諸如路由器之專用電腦。此等裝置可發現終端節點之間的路線，該等裝置可經由該等路線轉遞已分解成資料封包之訊息。可向每一節點指派獨特或全域位址，諸如網際網路協定(IP)位

103.12.23
年月日修正替換頁

址。網際網路為熟知的全球網際網路(inter-network)，其中電腦網路經由路由器彼此連接。

電腦網路協定具有分層架構。通常，最上層包括藉由諸如Web瀏覽器之應用程式提供的功能性。至少在終端節點中，此層可經由網路起始兩個電腦之間的連接。因此，例如，使用者可在其電腦上選擇所要網站。Web瀏覽器(在該電腦中執行)啟動程序，該程序導致將進行與相關聯於選定網站之伺服器的連接。Web瀏覽器經由一系列函式「向下」發送請求，該等函式被稱為網際網路協定套件或輸送控制協定/網際網路協定(TCP/IP)堆疊。此協定堆疊在其較高層處通常以時常作為在客戶端裝置中執行之作業系統(OS)程式之部分的軟體來實施。一旦已將選定網站轉譯成Web伺服器之IP位址，便經由網際網路來聯繫伺服器，且與在Web伺服器中實施之類似協定套件的上層程式進行適當連接。

為了使用該連接，使用者電腦中之TCP/IP堆疊囊封來自Web瀏覽器之請求訊息，在此實例中，該請求訊息為識別Web網頁之請求。可藉由沿著協定堆疊向下之若干垂直層(包括網路存取層)將訊息囊封一次以上。訊息最終到達客戶端裝置之最低層，即實體層(通常將其視為網路存取層之部分)。

在離開使用者電腦之實體層且接著前進通過網路中之一或多個躍點之後，來自Web瀏覽器之訊息到達Web伺服器，且「沿著」Web伺服器中之協定堆疊傳遞至視為Web瀏覽器之同級程式的程式。同級程式可接著藉由以下步驟對訊息作出回應：收集所請求Web網頁之資料且經由現有網路連接將該資料發送回至使用者電腦。將資料分解成多個訊息或封包，且以類似於發送請求訊息之方式的方式發送資料。

應用程式可具有藉由使用者客戶端電腦中之一或多個處理器執行的若干應用程式或處理程序。每一個別應用程式可產生不同類型之

網路資料訊務，其可具有不同封包損失、延時及流彈性要求。藉由實例，社會網路連接應用程式可經由網路傳達控制資料、文字、音訊及視訊，控制資料、文字、音訊及視訊中之每一者具有關於以上變數之不同要求。每一應用程式通常具備其自身之埠或埠群組以傳達此資料，但該等埠可皆共用使用者電腦中之相同較低層網路資源。在當前實施中，經由網路將每一客戶端裝置互連至特定目的地節點(亦即，另一客戶端或伺服器)之路由器包括大型傳輸及接收緩衝器。因而，存在很少封包損失或無封包損失，且通常准許客戶端裝置傳輸封包而不考慮流量控制，從而導致路由器佇列內之「緩衝器膨脹」。諸如TCP之協定為自調式協定，其判定壅塞且基於偵測到之封包損失來修改傳輸速度。當使用大型緩衝器來減輕封包損失時，TCP協定。

另外，在當前客戶端側實施中，封包之緩衝在驅動程式層級處發生。TCP/IP堆疊簡單地將封包下推至驅動程式，且驅動程式管理其自身之傳輸及接收佇列。因為在客戶端內之驅動程式層級處執行大量緩衝(乙太網路驅動程式可在封包傳輸之前緩衝佇列中多達4000個封包)，所以網路連接堆疊並不具備準確的網路/壅塞資訊。因而，需要在驅動程式層與網路堆疊層之間使用的用於在客戶端裝置內執行網路佇列處理之更智慧機制，其中利用回饋頻道。

【發明內容】

一種用於在一客戶端裝置上管理封包排程之電腦實施方法。舉例而言，方法之一項實施例包含：接收待傳輸之一封包；在一網路堆疊層級處將該封包排入一佇列中；判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。

【圖式簡單說明】

在隨附圖式之諸圖中藉由實例而非藉由限制來說明本發明之實施例，在該等圖式中，相似參考指示類似元件。應注意，在本發明中對本發明之「一」或「一個」實施例的參考未必為對同一實施例之參考，且其意謂至少一個。

圖1A至圖1B說明根據本發明之實施例的具有網路連接堆疊之客戶端裝置的方塊圖。

圖2A至圖2B說明根據本發明之兩種方法。

圖3A至圖3B說明根據本發明之兩個不同實施例的用於傳輸資料封包之在網路連接堆疊與驅動程式之間的通信。

圖3C說明根據本發明之一項實施例的客戶端側軟體架構。

圖4說明例示性發送者執行緒及例示性啟動者執行緒。

圖5說明例示性類別併列執行個體及排程器。

圖6說明服務類別映射至併列執行個體之一項實施例。

圖7說明服務類別映射至併列執行個體之另一實施例。

圖8說明通訊端訊務類別映射至服務類別之實施例。

圖9A說明具有內建式QFQ排程器組態之一項實施例中之鏈路共用分佈的最差情況。

圖9B說明根據一項實施例之封包篩選器(PF)組態的實例。

圖10A說明根據一項實施例之使用流雜湊的流量控制。

圖10B說明一項實施例中藉由併列處理演算法使用之當前分格集合及陰影複製(shadow)分格集合。

圖11A至圖11B說明根據本發明之兩個不同實施例的用於接收資料封包之在網路連接堆疊與驅動程式之間的通信。

圖12說明例示性工作迴圈執行緒、輪詢者(poller)執行緒及DLIL輸入執行緒。

圖13說明根據一項實施例之例示性執行緒資料集。

圖14說明用於一項實施例中之應用程式設計介面(API)。

圖15說明具有用於一項實施例中之API的複數個服務。

圖16說明客戶端側資料處理裝置之一項實施例。

圖17說明客戶端側資料處理裝置之另一實施例。

【實施方式】

本發明之實施例係有關用於作用中併列管理之電腦實施方法，該作用中併列管理用於在客戶端裝置上執行之網路連接應用程式。

圖1A為在上面可實施本發明之實施例的客戶端計算裝置101之方塊圖。所說明客戶端裝置101執行複數個應用程式105至107，該等應用程式經由網路150(諸如，網際網路)與伺服器120至121及其他客戶端裝置122通信。伺服器120至121可包括(藉由實例且非限制)Web伺服器、電子郵件伺服器、即時訊息傳遞伺服器，及檔案伺服器。在一項實施例中，應用程式105至107呼叫藉由網路連接堆疊102曝露之網路連接應用程式設計介面(API)208，以存取藉由網路連接堆疊102提供之網路連接資源。

此實施例之網路連接堆疊102包括併列管理邏輯115，該併列管理邏輯用於管理代表應用程式105至107中之每一者的複數個網路連接併列110至112。網路連接堆疊中之封包排程器116基於封包種類(如下文更詳細描述)來對待傳輸至併列中之每一者/自併列中之每一者接收之封包進行排程。儘管在圖1A中說明為單獨模組，但應瞭解，併列管理邏輯115及排程器116可實施為單一整合式軟體模組。

在一項實施例中，藉由併列管理邏輯110至112管理之併列110至112中的每一者包括：發送併列，其用於儲存傳出網路封包(例如，TCP/IP封包)；及接收併列，其用於儲存送入網路封包。基於應用程式105至107中之每一者的網路連接要求，針對應用程式105至107中之

每一者以不同方式管理發送及接收佇列。舉例而言，不同應用程式可具有不同的封包損失、延時及流彈性要求，前述各者之全部藉由佇列管理邏輯115監視及管理。在一項實施例中，提前(例如，在應用程式最初經由API 108註冊時)指定每一應用程式105至107之網路連接要求，且藉由佇列管理邏輯115基於指定要求來管理該應用程式之網路連接封包。藉由實例，Web瀏覽應用程式相較於即時視訊聊天應用程式通常更容許延時。因此，Web瀏覽應用程式將與不同佇列相關聯，該佇列具有不同於即時視訊聊天應用程式之指定服務等級。

在一項實施例中，安裝於客戶端裝置上之每一驅動程式150包括用於管理驅動程式佇列151至153之集合的佇列管理邏輯155，驅動程式佇列151至153中之每一者亦可與不同服務等級相關聯。此外，每一驅動程式可具有其自身之用於在驅動程式層級執行封包排程的排程器160，及其自身之用於管理佇列151至153之佇列管理邏輯155。如同網路連接堆疊102一樣，驅動程式佇列管理邏輯155及驅動程式排程器160可實施為單一邏輯模組(而非如在圖1中所說明之單獨模組)。在一項實施例中，每一驅動程式150可選擇獨自管理封包排程(本文中稱為「驅動程式管理」排程)，或可依賴於網路連接堆疊102之封包排程器116及佇列管理邏輯115來進行封包排程/佇列處理。

藉由實例且非限制，乙太網路驅動程式及蜂巢式無線(例如，3G、4G)驅動程式可依賴於藉由網路連接堆疊102之排程器116提供的封包排程及佇列處理，而802.11n(Wi-Fi)驅動程式可使用驅動程式排程器160來管理封包排程及佇列處理。在一項實施例中，Wi-Fi驅動程式實施無線多媒體延伸(WME)(亦稱為Wi-Fi多媒體(WMM)標準)，以根據四個優先權等級來對網路訊務進行排程，該四個優先權等級為語音、視訊、最佳努力及背景。然而，本發明之基本原理並不限於任何特定網路連接標準。

在一項實施例中，Wi-Fi驅動程式能夠在驅動程式管理之排程與網路層處之排程之間動態地切換。舉例而言，當在支援WME之802.11n網路上時，驅動程式可選擇驅動程式層級排程，但當在802.11b或802.11g網路上時，驅動程式可選擇網路層處之排程。在一項實施例中，當利用網路堆疊管理之排程時，網路堆疊102將向驅動程式通知封包何時準備好移出佇列。驅動程式將接著使封包移出佇列，且傳輸封包(如下文更詳細描述)。

如圖1A中所說明，與將封包自網路連接堆疊推送至驅動程式且在驅動程式中緩衝而與網路條件無關的先前實施形成對比，在本發明之一項實施例中，在驅動程式層150與網路連接堆疊102之間提供連續回饋171。自驅動程式150至網路連接堆疊之回饋確保：網路連接堆疊102知曉藉由驅動程式150管理之通信鏈路的網路連接條件，且可基於此知識執行封包排程/佇列處理。在一項實施例中，基於偵測到之網路連接條件，網路連接堆疊102智慧地實施封包排程及佇列處理，如本文中所描述。類似地，自網路連接堆疊102至驅動程式150之回饋信號向驅動程式通知網路連接堆疊之傳輸/接收佇列112內的條件(例如，新封包何時準備好自特定佇列進行傳輸)。

如圖1B中所說明，對於一些通信頻道，可在網路堆疊層處使用排程，且對於其他通信頻道(例如，如上文所論述之某些WiFi頻道)，可在驅動程式層處使用排程。詳言之，在所說明實施例中，對於至伺服器120及123以及客戶端121之通信頻道，可在網路堆疊層102中執行封包排程，而對於至伺服器122之通信頻道，可執行驅動程式管理之排程。此外，如圖1B中所展示，單一應用程式105可具有指派至不同佇列之不同類型之資料訊務，該等不同佇列支援不同封包損失、延時及流彈性要求。舉例而言，特定應用程式可開啓TCP/UDP通訊端以傳達控制資料、文字、音訊及視訊，前述各者中之每一者具有關於以上

變數之不同要求。因而，可將一類型之資料(例如，控制資料)排入與第一服務類別相關聯之佇列113中，且可將第二類型之資料(例如，互動式視訊)排入與第二服務類別相關聯之佇列110中。另外，不同應用程式可將資料排入與同一服務類別相關聯之同一佇列中。舉例而言，應用程式105及106可將控制資料排入與控制資料之服務類別相關聯的佇列110中，且應用程式106及107可將互動式視訊資料排入與互動式視訊資料之服務類別相關聯的佇列111中。

另外，應理解，取決於網路連接性(例如，客戶端101耦接至乙太網路抑或Wifi)及其他網路變數，客戶端裝置101可利用僅網路層佇列管理及/或排程或僅驅動程式管理之佇列管理及/或排程，同時仍遵照本發明之基本原理。

在圖2A中說明根據本發明之一項實施例的方法。在201處，在客戶端裝置上於協定堆疊之網路連接層處接收待傳輸之封包。若在202處判定該封包與使用驅動程式管理之排程的網路鏈路相關聯，則在203處，將封包提供至驅動程式層。在204處，驅動程式層接著將封包排入佇列、排程及傳輸。然而，若該封包與在網路層處執行封包排程之網路鏈路相關聯，則在205處，網路堆疊將封包排入佇列並排程以供傳輸。在206處，向驅動程式通知封包何時準備好傳輸，且在207處，驅動程式傳輸封包。

客戶端計算裝置101可為桌上型電腦、筆記型或膝上型電腦、視訊遊戲機，或其他消費型電子裝置。在本文中描述之些實施例中，客戶端裝置101為可包括以下功能之攜帶型無線裝置：雙向語音及視訊功能、電子郵件訊息傳遞功能，及媒體播放功能。在此實例中，客戶端裝置101與伺服器之間的通信路徑具有在客戶端裝置101與無線基地台(例如，小區塔台或Wifi存取點)之間的無線區段。在網際網路參考模型中，網路連接堆疊102、客戶端裝置101根據任何合適的無線通

信網路存取協定經由基地台與網路存取閘道器通信，下文給出無線通信網路存取協定之些實例。可經由另一基地台與閘道器之組合到達另一客戶端裝置122。以下各層在網路存取層之上：網際網路連接層(例如，定義用於網路上之每一節點的網際網路協定(IP)位址)、輸送層(例如，執行主機間流量控制以及開啓及關閉連接之輸送控制協定TCP)，及應用程式層(例如，諸如HTTP、SMTP及SSH之應用程式及處理程序協定)。

圖2B說明本發明之實施例，其中使用驅動程式管理之排程或網路堆疊管理之排程來傳輸封包。在251處，藉由特定應用程式經由開放通訊端連接來產生待傳輸之封包。舉例而言，互動式視訊應用程式可產生待傳輸至另一客戶端之視訊封包。在252處，基於封包之類型，網路層依據指定服務類別將封包排入佇列。舉例而言，如下文所論述，可針對10種不同類型之資料訊務來定義10個不同的服務類別來將資料排入佇列。因此，若封包為互動式視訊封包，則可將該封包排入用於互動式視訊之服務類別佇列中。類似地，若封包含有控制資料，則可將該封包排入用於網路控制之服務類別佇列中。

無論將封包排入佇列之方式，取決於驅動程式管理之排程抑或網路堆疊管理之排程而以不同方式將封包移出佇列。對於在254處判定之驅動程式管理之排程，在255處，驅動程式自指定服務類別執行移出佇列操作。舉例而言，若驅動程式正實施802.11n，則驅動程式可選擇使用藉由WMM定義之四個服務類別來執行排程(參見(例如)圖7，其說明服務類別與佇列執行個體之間的10:4映射)。或者，對於其他網路介面類型(例如，乙太網路、3G等)，可在網路層處執行排程(參見(例如)圖6，其說明服務類別與佇列執行個體之間的1:1映射)。因此，在260處，網路層自選定服務類別執行移出佇列操作。在270處，將封包提供至驅動程式層，在271處，該驅動程式層傳輸封包。

因此，自以上內容可看出，在一項實施例中，當需要傳輸封包時，將封包傳遞至針對網路介面經組態之網路層排程器。排程器自封包提取服務類別；服務類別判定供封包排入佇列上之佇列執行個體。接著將封包排入至對應佇列執行個體上；若佇列為滿的或正進行流量控制，則可丟棄封包(丟棄/排入佇列之決策留待如下文所描述之佇列處理規則/演算法(例如，SFB)來作出)。向驅動程式通知存在待執行之工作。在某個時間，驅動程式使封包移出佇列。需要識別佇列執行個體。若介面係針對「網路堆疊排程」進行組態，則排程器選擇待服務之適合佇列。若介面係針對「驅動程式排程」進行組態，則驅動程式向排程器指示待選擇以供服務之佇列。一旦識別出佇列執行個體，則自佇列移出封包(若可用)。接著將封包交遞至驅動程式以經由傳輸封包之媒體來傳輸。

如在圖1A至圖1B中所指示，在一項實施例中，將連續回饋172自網路連接堆疊102提供至應用程式105至107中之每一者(如藉由點線箭頭所指示)，且將連續回饋172用以提供對至/自應用程式105至107中之每一者之網路流的流量控制。舉例而言，當用於特定TCP或UDP通訊端之傳輸佇列110至112已達到指定臨限值時，產生回饋信號以指導各別應用程式105至107暫時中止或減少新封包傳輸。

圖3A至圖3B說明根據本發明之不同實施例的不同網路驅動程式模型。在圖3A中，應用程式301將待傳輸之封包發送至網路堆疊302(1)，網路堆疊302接著將網路封包發送至驅動程式之IO網路連接介面303(2)。在一項實施例中，IO網路連接介面303對封包進行分類，且基於封包種類將經分類之封包置放於適當IO輸出佇列304中(3)。如上文所提及，對於WMM，種類可包括語音、視訊、最佳努力及背景。驅動程式305接著使用其自身之封包排程器將封包自適當IP輸出佇列304移出(4、5)。

在說明於圖3B中之網路堆疊管理模型中，應用程式301將待傳輸之封包發送至網路堆疊302(1)，網路堆疊302接著對封包進行分類(使用下文所描述之分類方案)且將經分類之封包置放於適當發送佇列306中(2)。在一項實施例中，可存在與封包種類一樣多之不同發送佇列306(例如，對於10個不同封包種類，存在10個不同發送佇列)。網路連接堆疊302向驅動程式層通知(3)新封包何時準備好在佇列中之一者中傳輸。驅動程式307之IO網路連接介面接著將封包移出佇列，且將移出佇列之封包傳遞至驅動程式305以供傳輸(5、6)。

圖3C說明網路連接層102之額外架構細節，該網路連接層102包括：封包分類器202，其用於對封包進行分類；API 203，其用於與應用程式201、網際網路封包層206、輸送層205(例如，TCP、UDP)及通訊端層204介接；封包排程器209，其用於對封包傳輸進行排程；複數個類別佇列210；流量警示模組207；及核心程式設計介面(KPI)211。下文更詳細描述此等組件中之每一者。

A. 核心程式設計介面

在一項實施例中，使用以下私用KPI之集合：

`ifnet_allocate_extended()`

分配支援新輸出模型之ifnet執行個體。此執行個體為公用`ifnet_allocate()` KPI之延伸(私用)版本，其需要由呼叫程式來填充新定義之`ifnet_init_eparams`結構。此結構類似於`ifnet_init_params`，其中若干新欄位係關於新輸出模型：

ifnet_init_eparams欄位	描述
<code>pre_enqueue()</code>	若定義，則針對介面之每一傳出封包調用此回呼函式。驅動程式可對(完全形成之)封包執行最後改變，但在完成後驅動程式即負責呼叫 <code>ifnet_enqueue()</code> 以將封包排入佇列。(一些驅動程式將不需要註冊此回呼函式，但提供此回呼函式以防萬一。)
<code>start()</code>	此回呼函式用以向驅動程式指示：可藉由呼叫 <code>ifnet_dequeue()</code> 或 <code>ifnet_dequeue_multi()</code> 將一或多個封

	包移出佇列。當呼叫ifnet_start()時，調用此常式；將在專用核心執行緒之內容脈絡內執行該常式，因此保證該常式經單一執行緒化。若將直接自另一內容脈絡呼叫此回呼常式，則驅動程式必須使用額外串列化，以便防止競態條件相關問題。
output_sched_model	此欄位向網路連接堆疊告知驅動程式之封包排程要求。在預設情況下，網路連接堆疊將針對該介面選擇排程器及排程策略。該情形意謂當驅動程式將封包移出佇列時，移出佇列決策受排程器控制；此係「正常」排程模式。或者，驅動程式可選擇執行其自身之排程。在此「驅動程式管理」排程模型中，網路堆疊簡單地提供佇列執行個體，且接著驅動程式需要指定佇列，將自該佇列移出封包。後一模型之使用狀況實例為併有Wi-Fi多媒體(WMM)之802.11驅動程式；硬體結合存取點執行封包排程。
sndq_maxlen	輸出佇列之最大大小；此大致為來自驅動程式之關於其傳輸佇列之大小的提示。當sndq_maxlen設定為0時，網路堆疊將試圖拾取合理大小(當前設定為128個封包。)
output_ctl()	提供網路連接堆疊向基本驅動程式/程式系列告知輸出參數(例如，硬體有效負載、傳輸環大小等)之改變的方式，以供未來使用
{input_bw, input_bw_max}	有效及理論輸入/下行鏈路速率；出於資訊目的。
{output_bw, output_bw_max}	有效及理論輸出/上行鏈路速率。

ifnet_enqueue()

將封包排入實施新驅動程式輸出模型之介面的輸出佇列中。針對實施pre_enqueue()回呼函式之驅動程式/程式系列提供此ifnet_enqueue()。

{ifnet_dequeue, ifnet_dequeue_multi}()

將一或多個封包自實施新驅動程式輸出模型之介面的輸出佇列移出，且將排程模式設定為「正常」。

{ifnet_dequeue_service_class, ifnet_dequeue_service_class_multi}()

將一或多個封包自實施新驅動程式輸出模型之介面的輸出佇列移出，且將排程模式設定為「驅動程式管理」。

ifnet_set_output_sched_model()

將排程模式設定為「正常」或「驅動程式管理」。

{ifnet_set_sndq maxlen, ifnet_get_sndq maxlen}()

設定並獲得輸出佇列之最大長度。

ifnet_get_sndq_len()

獲得輸出佇列之當前長度。

ifnet_start()

在實施新驅動程式輸出模型之介面上觸發驅動程式層處之傳輸。若尚未調用驅動程式之start()回呼函式，則此觸發可導致調用該start()回呼函式。

{ifnet_set_bandwidths, ifnet_bandwidths}()

設定並獲得介面之上行鏈路及下行鏈路速率。無論何時資訊在驅動程式層處可用，該等速率便可在ifnet連結(attach)之後藉由驅動程式在任何時間來設定。

{ifnet_transmit_burst_start, ifnet_transmit_burst_end}()

當驅動程式不能容易地自硬體擷取上行鏈路速率時用來估計此資訊之替代機制。此等資訊向網路連接堆疊告知叢發之傳輸的開始及結束。

在一項實施例中，用IFEF_TXSTART旗標來為自身已註冊為支援新輸出模型(亦即，網路堆疊管理之排程)之驅動程式加旗標。

B. 資料鏈路介面層(DLIL)208

啓動者執行緒402(說明於圖4中之一項實施例)

支援新輸出模型(亦即，網路層排程)之介面使用專用核心執行緒，即「啓動者執行緒」402，該專用核心執行緒之工作為調用驅動程式之start()回呼函式。在一項實施例中，無論何時呼叫ifnet_start()，若尚未執行此啓動者執行緒，則用信號通知此執行緒作為經由ifnet_enqueue()將封包排入佇列之部分來執行，從而允許應用程式執行緒在將封包排入至輸出佇列後立即返回。此執行緒提供驅動

程式之start()回呼函式之串列化形式，使得移出佇列可按次序發生。此執行緒亦降低驅動程式層處之複雜度，此係因爲驅動程式可不用過多地擔心影響來執行可瞬間阻斷執行緒執行之某些操作(硬體相關或非硬體相關)，當網路連接堆疊自此啓動者執行緒之內容脈絡執行驅動程式之start()回呼函式時，網路連接堆疊並不保持鎖定，因此可瞬間阻斷執行緒執行。

符記桶調節器

另外，當符記桶調節器(TBR)經組態用於介面時，網路層管理之輸出模型實現ifnet層處之一種形式的上行鏈路速率限制。在預設情況下，介面並不具有經組態之TBR；使得TBR需要經由ifconfig(8)或pfctl(8)來手動組態。當啓用TBR時，無論何時輸出佇列爲非空的(在圖4中之402處所說明)，啓動者執行緒便將週期性地喚醒(每10 ms一次)。在每一週期期間，允許驅動程式將與可用符記一樣多之位元組移出佇列；在每一週期開始時再填充符記。根據藉此組態TBR之速率來計算符記之數目。一個特定TBR實施不需要分配呼出(不同於BSD採用之方法)；因爲此情形，該特定TBR實施可在具有極低CPU額外耗用之情況下適應極高速率(數十Gbps)，此係因爲時間間隔固定且因此獨立於呼出解析度(跨越不同平台可達成10 ms之時間間隔)。

傳輸佇列(說明於圖5中之一項實施例)

ifnet之if_snd成員保持介面之傳輸佇列。此資料結構含有關於內建式排程器(類型、執行個體、回呼函式)、TBR及(視情況)替代排程器之資訊。

在預設情況下，在一項實施例中，系統建立封包排程器之內建式執行個體(ifcq執行個體)。如所提及，封包排程器及其參數之選擇取決於網路介面之類型，且在某一例子中亦取決於網路之拓撲。在一項實施例中，當連結封包排程器時，該封包排程器將其執行個體儲存



於ifcq_disc中，且將enqueue()、dequeue()及request()回呼函式組態至排程器之對應常式。對於需要「驅動程式管理」模型之介面，連結專用排程器，該專用排程器提供替代dequeue_sc()而非dequeue()回呼函式。此等回呼函式之某些實施例如下：

排程器函式	描述
ifclassq_enq_func()	使單一封包排入佇列至排程器。可能的傳回類型為：在將封包成功排入佇列時，傳回CLASSSEQ_SUCCESS；在將封包成功排入正確證流量控制之佇列上時，傳回CLASSSEQ_SUCCESS_FC；在丟棄封包時，傳回CLASSSEQ_DROPPED；在因為佇列超出其流量控制限值而丟棄封包時，傳回CLASSSEQ_DROPPED_FC；及在因為暫時中止佇列而丟棄封包時，傳回CLASSSEQ_DROPPED_SP。
ifclassq_deq_func()	自排程器將單一封包移出佇列；此係排程器判定為最適合傳輸之任何封包。移出佇列作業碼可為用於正常移出佇列之CLASSDQ_REMOVE，或用於擷取適合封包而實際上不將封包移出佇列的CLASSDQ_POLL。
ifclassq_deq_sc_func()	類似於常規移出佇列回呼函式，惟呼叫程式判定適合傳輸之封包的服務類別除外。
ifclassq_req_func()	至排程器之請求回呼函式。最後參數取決於作出之請求之類型。該等類型為：CLASSQRQ_PURGE，其用於去除排程器中之所有佇列；CLASSQRQ_PURGE_SC，其用於去除特定佇列或視情況該佇列內之流；CLASSQRQ_EVENT，其用於傳播介面事件(速度、MTU、鏈路狀態之改變)；及CLASSQRQ_THROTTLE，其用於應用該佇列之節流參數。

排程器116之一項實施例使N數目個類別執行個體化；每一類別與一服務類別相關且管理佇列執行個體110至112。取決於封包進行分類之方式，將封包排入此等佇列執行個體中之一者中。當組態PF_ALTO支援時，可經由封包篩選器(PF)基礎設施(例如，藉由

pfctl(8))更動內建式排程器及其參數。此情形提供使封包排程之不同特性模型化之便利方式(例如，試用不同排程器及/或參數)。

封包排程器 116

封包排程器模組 116 之一項實施例提供進入點，以將封包排入至其類別佇列執行個體 210 中之一者及自其類別佇列執行個體 210 中之一者移出。在一項實施例中，每一類別對應於一佇列。每一類別取決於排程演算法及參數而管理其所有佇列。

在一項實施例中，經由以下技術中之一者組態排程器 116 且將其連結至 ifnet：

1. **內建式(靜態)**：當將介面連結至網路連接堆疊時，基於驅動程式所請求之佇列排程模型來選擇排程器。對於「正常」模型，堆疊建立具有 10 個類別(因此 10 個佇列)之排程器。對於「驅動程式管理」模型，替代地建立具有 4 個類別之專用排程器的執行個體。

2. **PF(動態)**：在一項實施例中，可藉由經由 PF 框架組態排程器及其參數來更動內建式(靜態)組態。此更動需要啓用 PF_ALTQ 組態選項並呈現「altq=1」boot-args NVRAM 選項。在一項實施例中，在預設情況下不啓用/允許 PF_ALTQ 組態。然而，當啓用時，PF_ALTQ 組態實現用於藉由不同排程器及參數進行實驗之便利且有利的機制。

在一項實施例中，以下排程器在預設情況下不使用，且僅藉由 PF 可用：

排程演算法	描述
PRIQ	優先佇列處理。此為非階層式排程器。該排程器並不需要上行鏈路頻寬之任何知識。此排程器作為原始 ALTQ 程式碼之部分。
FAIRQ	公平佇列處理。此為非階層式排程器且為 PRIQ 之添加公平性的延伸。該排程器需要知曉上行鏈路頻寬，且每一優先佇列必須指派有一頻寬。

CBQ	基於類別之佇列處理。此為階層式排程器。該排程器需要知曉上行鏈路頻寬。
HFSC	階層式公平服務曲線(Hierarchical Fair Service Curve)。此為階層式排程器。該排程器需要知曉上行鏈路頻寬，且該排程器亦相當複雜但為獨特的，此係因為其同時支援即時服務、鏈路共用服務及速率限制服務。該排程器允許每一佇列組態有作為即時服務之部分的保證延遲特性。此排程器作為原始ALTQ程式碼之部分。

在一項實施例中，在預設情況下(亦即，在網路連接堆疊層102處)使用以下排程器：

排程演算法	描述
QFQ	快速公平佇列處理。此為非階層式排程器。該排程器不需要上行鏈路頻寬之任何知識，且極易組態。該排程器亦為提供公平性及頻寬分佈保證之彼等排程器中的最高效排程器。此排程器亦可經由封包篩選器(PF)來組態。在一項實施例中，QFQ用於網路層管理之排程。QFQ演算法之一項實施例描述於題為 QFQ: Efficient Packet Scheduling With Tight Bandwidth Distribution Guarantees 之文獻中，該文獻係作為附錄A而附加至本專利申請案。
TCQ	訊務類別佇列處理。此為非階層式被動排程器。該排程器不使用任何排程演算法，且簡單地提供服務類別與群組之間的映射。此排程器依賴於驅動程式或硬體來執行實際封包排程(亦即，驅動程式管理之排程)，此係因為驅動程式負責選擇適合傳輸之服務類別。

映射

1. 1:1映射

如圖6中所說明，在一項實施例中用於網路層管理之排程的QFQ組態分別提供封包服務類別601至610與封包佇列執行個體611至621之間的1:1映射。如所說明，將10個服務等級粗略地分成4個群組630、640、650、660，且在每一群組內提供優先順序。依據以下各者基於

經分類之訊務的特性來定義群組：延遲容限(低至高)、損失容限(低至高)、彈性對非彈性流，以及諸如封包大小及速率之其他因數。如本文中所描述，「彈性」流為需要相對固定頻寬之流，而「非彈性」流為可接受非固定頻寬之流。所說明1:1映射允許網路連接堆疊達成對每一服務類別之特性(behavior)及區分的完全控制：根據封包進行分類之方式來將封包直接排入至佇列中之一者中；在移出佇列期間，排程器判定將自最適合佇列傳輸之封包。

2. 10:4映射

如圖7中所說明，在一項實施例中用於驅動程式管理之排程的TCQ組態提供10個服務類別601至610與4個佇列執行個體701至704之間的10:4映射。在排程器並不執行任何一種封包排程而是簡單地提供佇列執行個體並將服務類別映射至佇列的意義上，此排程器為被動的。因為此實施例之網路連接堆疊不控制排程，所以佇列不可定義有類似於1:1映射之彼等特性的特性。實情為，可感知佇列為具有範圍為低(L)至最高(H+)之優先權。將封包直接排入至佇列中之一者中，該佇列表示最初藉以對封包進行分類之服務類別。在移出佇列期間，驅動程式負責選擇最適合傳輸之服務類別。將佇列之數目設定為4，且其為實施人工因素；如所提及，此亦恰巧為802.11 WMM存取種類之數目。

佇列處理規則

在一項實施例中，佇列處理規則或演算法模組管理類別佇列之單一執行個體；佇列簡單地由一或多個封包(mbuf)組成。演算法負責判定應將封包排入佇列或丟棄。

在一項實施例中，經由以下方式中之一者組態佇列處理演算法且將其連結至排程器類別之執行個體：

1. 內建式(網路堆疊管理式)：當作為在ifnet上組態封包排程器



之部分來將排程器類別執行個體化時，選擇佇列處理演算法。排程器之所有類別共用同一佇列演算法(每一類別具有其自身之獨特執行個體)。

2. PF(動態或驅動程式管理式)：或者，可藉由經由封包篩選器(PF)框架組態排程器及其參數(包括依據類別之佇列處理演算法)來更動內建式(靜態)組態。

演算法

在一項實施例中，以下佇列處理演算法在預設情況下不使用，且僅經由PF可用：

佇列處理演算法	描述
DROP-TAIL	尾部丟棄之基本FIFO。
RED	隨機早期丟棄。此演算法來自ALTQ。
RIO	使用IN/OUT位元之RED。此演算法來自ALTQ。
BLUE	BLUE維持丟棄/標記機率。當佇列超出其分配量時，使機率增加。當佇列為空時，使機率減小。當機率達到某一限值時，丟棄封包/對封包作出標記。不同於RED，此演算法需要極少調節或不需要調節；然而，此演算法不能追蹤多個流。
SFB	推測公平BLUE，本質上為BLUE之公平變體，其基於流是超出其分配量抑或處於閒置而雜湊流，且針對每一流維持不同標記/丟棄機率。用於一項實施例中之SFB演算法描述於題為 Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairnesss 之文獻中，該文獻係作為附錄B而附加至本專利申請案。SFB亦提供用於非彈性流之額外速率限制功能性。SFB追蹤多個流之能力對於流量警示機制(以及自流量警示機制導出之彼等機制)為重要的。此為藉由本專利申請案之受讓人設計之SFB演算法的原始實施/解譯。

封包分類

如所提及，在一項實施例中，將每一傳出封包排入對應於封包之服務類別的類別佇列執行個體中。服務類別指派或封包分類發生於

整個網路連接堆疊內之若干處。一般而言，封包分類可為顯式的(由應用程式決定採用)或隱式的(藉由系統設定或更動)。

顯式分類：在一項實施例中，使用在圖8中說明為映射至服務類別之以下訊務服務類別值中之一者，藉由經由setsockopt(2)以固著方式或藉由sendmsg(2)在每訊息基礎上發出SO_TRAFFIC_CLASS選項，應用程式可對其訊務進行分類：

訊務服務類別	描述
SO_TC_BK_SYS	「背景系統起始」，高延遲容限、高損失容限、彈性流、可變大小與長使用期限。例如，系統起始之iCloud™同步或時間囊(Time Capsule)備份，對於iCloud™同步或時間囊備份，不存在進度回饋。
SO_TC_BK	「背景」，使用者起始，高延遲容限、高損失容限、彈性流、可變大小。例如，使用者起始之iCloud同步或時間囊備份；或背景應用程式之訊務，對於背景應用程式，存在某一進度回饋。
SO_TC_BE	「最佳努力」，未分類/標準。在一項實施例中，此類別為預設服務類別。
SO_TC_RD	「回應性資料」，高於「最佳努力」之識別記號(notch)，中等延遲容限、彈性與非彈性流、叢發性、長使用期限。例如，電子郵件、即時訊息傳遞，對於前述兩者，存在互動性及緊急性意義(使用者等待輸出)。
SO_TC_OAM	「操作、系統管理及管理」，中等延遲容限、低至中等損失容限、彈性與非彈性流、可變大小。例如，虛擬專用網路(VPN)通道。
SO_TC_AV	「多媒體音訊/視訊串流傳輸」，中等延遲容限、低至中等損失容限、彈性流、恆定封包時間間隔、可變速率與大小。例如，AirPlay™播放(視訊及音訊兩者)。
SO_TC_RV	「回應性多媒體音訊/視訊」，低延遲容限、低至中等損失容限、彈性流、可變封包時間間隔、速率及大小。例如，AirPlay鏡像、螢幕共用。
SO_TC_VI	「互動式視訊」，低延遲容限、低至中等損失容限、彈性流、恆定封包時間間隔、可變速率及大小。例如，FaceTime™視訊。
SO_TC_VO	「互動式語音」，低延遲容限、低損失容

	限、非彈性流、恆定封包速率、某種程度上固定之大小。例如，包括FaceTime™音訊之VoIP。
SO_TC_CTL	「網路控制」，低延遲容限、低損失容限、非彈性流，速率為叢發性但短的、可變大小。例如，域名服務(DNS)查詢；某些類型之本端發起之網際網路控制訊息協定(ICMP)、ICMPv6；網際網路群組管理協定(IGMP)/多播收聽者發現(MLD)加入/離開、位址解析度協定(ARP)。TCP/UDP確認(ACK)。

因此，自上文可看出，在一項實施例中，系統將網路控制封包指派給最高優先權種類，藉此確保控制封包先於具有所有較低種類之封包而轉遞。此情形為優於如下先前系統之改良：其中某些控制封包(例如，TCP確認(ACK)封包)可在佇列中阻塞在其他類型之非控制封包之後(藉此降低系統效能)。

在一項實施例中，在語音通話正發生於客戶端裝置上時，將暫時中止在佇列中分類為背景系統起始(SO_TC_BK_SYS)之任何封包。因而，此實施例提供優於如下先前系統之顯著益處：其中可由於正傳輸低優先權封包(例如，背景系統起始之封包)而使語音通話降級或中斷。因此，在此實施例中，待備份至服務(例如，iCloud)之使用者相片串流或其他資料將不干擾語音通話。

系統之一項實施例可阻止將訊務標記為背景系統起始，使得訊務不干擾來電通話，藉此增加通話之可靠性。當起始通話時，網路層(例如，TCP/IP層)將接收暫時中止所有背景起始之訊務的流量控制通知。作為回應，網路層可停止將更多封包向下發送至介面。網路層亦可阻止應用程式沿著網路堆疊寫入更多資料。此情形將幫助改良CPU利用率，此係因為停頓了應用程式，且此情形亦改良語音通話之可靠性。若語音通話在合理持續時間內完成，則應用程式可重新繼續資料通信。

在一項實施例中，當暫時中止特定較低優先權應用程式時，網路堆疊102將(週期性地)探測通信鏈路(例如，經由回饋信號171)以判定其何時可重新繼續傳輸，且將把此資訊傳達至各別應用程式。當不再加載鏈路時，將接著重新繼續封包傳輸。

簡言之，驅動程式層150與網路連接堆疊102之間的連續流量控制回饋171及網路連接堆疊與應用程式105至107之間的回饋172提供對網路頻道及頻寬之更智慧、高效分配。

在一項實施例中，以上值並不意味是保證，而是自應用程式至系統之關於其訊務之特性的提示。系統將儘力基於訊務之種類來提供關於訊務之某種形式之區分，但歸因於在封包排程參數至網路拓撲或媒體條件範圍內之變化因數而無法作出保證。

在一項實施例中，藉由與以上值中之一者相關聯的通訊端產生之訊務將在併列/緩衝器(mbuf)中攜載對應服務類別值；如在圖8中所說明，在SO_TC值與MBUF_SC值之間存在1:1映射。

圖9A說明具有內建式QFQ排程器組態之一項實施例中鏈路共用分佈的最差情況。除通訊端層級分類外，在整個核心之某些位置使用MBUF_SC_CTL服務類別來對某些控制類型之封包進行分類(例如，ARP、ND NS/NA、IGMP/MLD)。此分類亦包括臨時提高由TCP ACK使用之服務類別。

隱式分類：此形式之分類可能經由封包篩選器(PF)框架實現。無關於IP訊務最初在發源處進行分類之方式，該分類允許經由PF安裝分類規則且對所有IP訊務生效。已藉由服務類別相關之關鍵字來增強PF及pfctl(8)；圖9B說明可藉由pfctl(8)處理以更動內建式設定之PF組態檔案的實例。

因此，在顯式分類狀況下，應用程式開啓具有預設服務類別(BE)之通訊端。應用程式可經由SO_TRAFFIC_CLASS通訊端選項來設定

通訊端之服務類別，使得所有未來的發送/寫入操作將自動使封包標記有對應服務類別。應用程式可選擇使沿著通訊端發送之每一資料報與SO_TRAFFIC_CLASS輔助訊息選項選擇性地相關聯，使得相關聯之封包將標記有對應服務類別(但將不影響其他當前或未來封包)。在此狀況下，可易於具有與此通訊端相關聯之許多不同服務類別。

在隱式分類狀況下，分類規則安裝於封包篩選器引擎中。每一規則含有簽章(例如，協定、埠等)，該簽章在匹配後便將導致封包標記有服務類別。

● 分類標籤。除用MBUF_SC值標記併列/緩衝器(mbuf)外，在一項實施例中，執行封包分類之模組202亦使一或多個標籤與封包相關聯，以便輔助系統之剩餘部分識別封包之類型或流。在一項實施例中，此等標籤駐存於mbuf之內建式pf_mtag子結構內，且無關於執行分類之方式(顯式或隱式)而經設定。在一項實施例中使用之標籤如下：

標籤	描述
PF_TAG_FLOWHASH	已計算流雜湊且其與mbuf相關聯。
PF_TAG_FLOWADV	封包屬於本端終止之流，其具有流量警示能力。
PF_TAG_TCP	最外封包使用TCP作為輸送協定。

流量回饋

● 流雜湊：如圖10A中所說明，在一項實施例中，正經由介面發送出之每一mbuf藉由流雜湊1001加標籤(因此標記有PF_TAG_FLOWHASH)，其將幫助識別屬於介面層處之特定流的所有封包。在一項實施例中，流雜湊1001為32位元整數，且在以下位置中之一者中計算該流雜湊1001：

1. 通訊端204。在一項實施例中，當連接通訊端時，計算並儲存通訊端204之流雜湊。此通訊端上之其他傳輸將使雜湊值攜載於封

包之mbuf結構內。

2. 封包篩選器(PF)。在一項實施例中，當封包進入驅動程式150時，除非已對流雜湊進行分類，否則以相關聯之PF規則及狀態來計算及儲存流雜湊。若將封包成功地傳遞回至IP 206，則在mbuf結構中，封包將攜載有流雜湊，該流雜湊相關聯於與封包匹配之規則或狀態。

在一項實施例中，取決於效能，用以計算流雜湊之雜湊演算法在多個計算系統平台上不同。下表說明例示性平台及對應雜湊：

平台	雜湊演算法
Intel 32位元	MurmurHash3(32位元，i386變體)
Intel 64位元	MurmurHash3(128位元，x86_64變體)
ARM等	32位元JHash

用於TCP之流量控制與警示：使用本文中所描述之佇列管理技術，當在介面處排入佇列之每個流之封包數目達到上限時，對使用TCP發送之應用程式進行流量控制。替代使用類似於顯式壅塞通知(ECN)或封包丟棄之指示符，介面將流量警示回饋提供至輸送層。可提供此回饋而無任何封包損失。

當以下兩個條件中之一者為真時，自AQM接收對連接之流量警示：

1. TCP連接之發送速率增加到超過鏈路上所支援之頻寬。
2. 自裝置之無線鏈路上的可用頻寬下降，該無線鏈路為第一躍點。

在此等兩個狀況中，發送更多封包將使封包累積於介面佇列中，且將使應用程式經歷之延時增加。否則，發送更多封包可能引起封包丟棄，封包丟棄將降低效能，此係因為TCP發送者將必須重傳輸彼等封包。藉由使用流量警示機制，TCP發送者可在不經受任何封包損失或任何效能損失之情況下適應於可用頻寬。介面佇列將從不丟棄

TCP封包，但其將僅將流量警示發送至連接。因為此機制，顯著減少裝置驅動程式中之緩衝量，從而導致裝置上所有TCP連接之延時得到改良。

TCP連接對流量警示之主要回應為減小其壅塞窗，此將實際上減小其發送速率。此情形藉由使緩慢啟動臨限值後移及允許連接進入壅塞避免階段來進行。但若連接已在恢復中，則其意謂連接在該往返時間中已經歷封包損失，且已減低了其發送速率。在此狀況下，不再進一步減小壅塞窗。

受到流量控制之連接將避免使通訊端可寫，直至流量控制提升為止。當網路堆疊不能在介面上發送出封包時，以上情形將防止應用程式寫入可能僅緩衝在網路堆疊中之更多資料。此情形將幫助需要僅發送最近更新且最好拋棄較舊更新的互動式應用程式。

在處於流量控制狀態時，若在以重複確認或SACK資訊之形式接收到之TCP確認中存在封包損失之指示，則連接將中止流量控制且啓動快速恢復以立刻重傳輸遺失資料。此情形將使延時不再增加，此係因為在恢復期間發送之封包的速率受到限制。由於使介面保證不丟棄任何TCP封包，因此連接將能夠儘可能快速地重傳輸遺失資料。

當連接處於流量控制狀態時，其意謂封包正以比先前速度慢之速度離開介面。在此情形下，準備好發送之介面佇列中可存在等待封包。通常，此等封包為在最後發送窗結尾之封包。若此等待時間大於關於連接而計算之重傳輸逾時，則逾時將激發。就此而言，重傳輸已發送之資料可在介面佇列中建立同一封包之複本。此建立稍後可產生重複確認，且使連接不必要的進入恢復。

為避免此混淆，受到流量控制之TCP連接將避免根據重傳輸計時器來重傳輸封包，直至稍後時間為止。若等待時間過長，則替代永遠等待，連接可變得逾時，且將使錯誤傳回至應用程式。

每當激發重傳輸逾時時，在再次嘗試之前使計時器後移。此情形將幫助偵測鏈路上延遲之急劇增加。但對於受到流量控制之通訊端而言，延遲可為臨時將流阻斷之結果。當連接脫離流量控制狀態時，取消後移。此情形將幫助此後及時地激發重傳輸計時器。

當介面佇列中之封包流出且佇列位準降至臨限值以下時，介面將產生流量警示以使受到流量控制之所有流再次開始發送資料。就此而言，TCP通訊端亦變為可寫，且應用程式可開始寫入資料。

當流量控制提昇時，連接將發送先前從未發送過之新資料。此情形將產生新確認，且將啓動ACK計時器。其亦將幫助偵測在流量控制之前是否存在尚未偵測到之任何資料損失。若不存在待發送之新資料，則重傳輸計時器將即刻激發，且重傳輸計時器將觸發尚未確認之任何未處理資料的重傳輸。

使用流量警示及流量控制機制，TCP連接將能夠適應於鏈路頻寬之變化，且將能夠最小化由在主機上之多個層級處緩衝封包所誘發之延遲。

用於UDP之流量控制與警示：在一項實施例中，僅在UDP通訊端使用connect()系統呼叫連接至同級UDP通訊端時，UDP通訊端才能夠進行流量控制。當介面佇列中來自UDP流之封包的數目大於流量控制之限值時，產生警示。此時，將通訊端標記為受到流量控制。不同於TCP，介面佇列將丟棄此後產生之所有UDP封包。通訊端將為不可寫入的，此意謂等待使用選擇或輪詢或kevent系統類別之寫入事件的應用程式將不能獲得事件，直至流量控制得到提昇為止。若應用程式無論以何種方式將資料寫入至通訊端，封包皆將被通訊端層丟棄，且將傳回錯誤(ENOBUFS)。

此情形不同於如下先前行為：其中所有UDP寫入繼續進行，但稍後藉由驅動程式丟棄封包。UDP流量控制及警示將向應用程式給出立

即回饋，使得應用程式可立即減小其發送速率。舉例而言，視訊應用程式可改變其編碼以經由網路發送較少資料。

由於在受到流量控制之UDP通訊端上之通訊端層處丟棄封包，因此相較於處理封包且始終將其發送至驅動程式而僅被丟棄的先前情形，此情形節省大量CPU利用率。另一優點為，受到流量控制之UDP流不會使(overwhelm)介面爆滿。此優點將減小引起的封包損失，且改良主機上其他流之延時。

在一項實施例中，歸因於使用推測公平Blue(SFB)(參見本專利申請案之附錄B)作為併列處理演算法，有可能追蹤介面層處之流。在一項實施例中，SFB之實施使用2階布隆(bloom)過濾器，藉此將流(如藉由流雜湊值1001所指示)映射至每一SFB層級1009處的恰好一個分格。此實施例之每一分格追蹤封包之數目以及流丟棄/標記機率。在一項實施例中，SFB亦追蹤正受到流量控制之流的清單。流量控制及流量警示之臨限值係基於分格分配量(當前設定為併列限值之1/3)。分格機率得到相應更新，但其當前不用於速率限制。

流暫時中止與重新繼續

在一項實施例中，當網路介面受到節流時，標記為「機會性」之某些通訊端被暫時中止。在一項實施例中，當將藉由此等通訊端產生之封包排入受影響併列上時，將丟棄該等封包。在一項實施例中，NOTE_SUSPENDED事件將產生於通訊端上，以便向應用程式告知無限期地阻斷通訊端上之訊務。應用程式可接著決定是否中止連接。當介面不再受到節流時，受影響併列將不再阻斷封包，且NOTE_RESUMED事件將產生於受影響通訊端上。在內部，同一機制可由流量控制及警示用於實施暫時中止及重新繼續。

傳入網路層排程模型

一項實施例之機會性輪詢使用如圖11B中所說明之網路驅動程式

輸入模型。驅動程式組件1111輪詢硬體1112以查詢送入封包，且IO網路連接介面1110輪詢驅動程式。每一接收佇列執行個體輪詢(3)驅動程式1111之IP網路連接介面1110，以判定是否存在與該接收佇列相關聯之任何封包，該等封包已準備好移出佇列(3)且向上傳遞至網路堆疊1108(4)且隨後向上傳遞至請求應用程式1107(5)(如在(6)處所說明，請求應用程式1107輪詢與其相關聯之接收佇列執行個體1109以查詢新封包)。

因此，藉由此新模型，傳入封包不再如圖11A中之操作1至6所說明般藉由驅動程式/程式系列向上推送至網路連接堆疊。實情為，傳入封包駐存於驅動程式之接收佇列1109中，直至傳入封包藉由網路連接堆疊1108移出佇列為止。在一項實施例中，此情形涉及關閉客戶端裝置硬體之接收中斷(IRQ)。所描述實施例為獨特的一個原因為，網路連接堆疊1108(結合驅動程式)取決於負載因數而在輪詢(圖11B)與舊版模型(圖11A)之間交替。在一項實施例中，當負載達到預定臨限值(例如，聚集於驅動程式之佇列中之封包的指定等級)時，系統可接著轉變至舊版模型(圖11A)。當轉變至舊版模型時，開啓硬體之接收IRQ，且驅動程式1105將封包自IO網路連接介面1104向上推送至適當接收佇列執行個體1103且最終經由網路堆疊1102推送至請求應用程式1101。

核心程式設計介面(KPI)

在一項實施例中，為了適應以上組態，使用私用KPI之集合：

`ifnet_allocate_extended()`

向支援新輸入模型之ifnet執行個體分配若干相關欄位：

ifnet_init_eparams欄位	描述
<code>input_poll()</code>	藉由網路堆疊1108呼叫此回呼函式，以自驅動程式1111擷取一或多個封包，驅動程式1111實施新驅動程式輸入模型。

<code>input_ctl()</code>	藉由網路堆疊1108調用此回呼函式，以向驅動程式1111告知及指導輪詢對舊版模型轉變。
<code>rcvq_maxlen</code>	驅動程式接收佇列之最大大小。

`ifnet_input_extended()`

類似於`ifnet_input()`，惟驅動程式1111向網路連接堆疊1108提供與以下各者相關之所有資訊除外：封包鏈之開始及結束，以及總封包及位元組計數。鼓勵已擁有此資訊之驅動程式利用此新變體，此係因為新變體實現更高效率。可無關於驅動程式是否採用新模型(圖11B)而使用此函式。

● 在一項實施例中，用`IFEF_RXPOLL`旗標來為自身已註冊為支援新輸入模型(圖11B)之驅動程式加旗標。

資料鏈路介面層(DLIL)208

輸入執行緒：在一項實施例中，在DLIL輸入執行緒之內容脈絡內發生整個網路連接堆疊內之輸入封包處理。一些介面具有其自身之專用輸入執行緒，而其他介面共用共同(主)輸入執行緒。在一項實施例中，存在DLIL輸入執行緒之3個變體：

1. 主輸入執行緒

● 在一項實施例中，由以下各者使用該主輸入執行緒：回送介面，以及並未獲得其自身之專用輸入執行緒的其他介面(亦即，除乙太網路/PDP或不支援RXPOLL之介面外的任何介面)。此執行緒亦用於處置所有協定註冊及封包注入。此執行緒以`dlil_main_input_thread_func()`來實施。

2. 舊版輸入執行緒

在一項實施例中，舊版輸入執行緒由不採用RXPOLL模型之乙太網路/PDP介面使用，以`dlil_input_thread_func()`來實施。

3. RXPOLL

在一項實施例中，RXPOLL由採用RXPOLL模型之任何介面使用，以dlil_rxpoll_input_thread_func()來實施。

輪詢者執行緒

在一項實施例中，支援新輸入模型(RXPOLL)之介面使用專用核心執行緒，即「輪詢者執行緒」(在圖12中之1202處說明)，該執行緒之工作為調用驅動程式之input_poll()回呼函式以便自驅動程式擷取一或多個封包；此操作在輪詢「開啓」時發生，否則輪詢者執行緒1202保持休眠。此執行緒類似於在圖12中之1201處說明之工作迴圈執行緒，其中該等執行緒皆結束呼叫ifnet_input()，以便將封包儲放至具有RXPOLL能力之DLIL輸入執行緒的接收佇列。

接著將封包沿著網路連接堆疊向上發送，以供自此DLIL輸入執行緒之內容脈絡進行進一步處理。

機會性輪詢

在一項實施例中，具有RXPOLL能力之介面在IFNET_MODEL_INPUT_POLL_OFF 模式與IFNET_MODEL_INPUT_POLL_ON模式之間轉變。在一項實施例中，前一模式為預設/初始模式；當網路堆疊判定負載因數為低時，網路堆疊為介面選擇此模式。當前藉由查看以下各者來判定負載因數：DLIL接收佇列中之封包及位元組的EWMA(P_avg, B_avg)，及DLIL輸入執行緒喚醒請求之EWMA(W_avg)。

參看圖12中之DLL輸入執行緒1203，在一項實施例中，當($P_avg \geq P_hiwat \&& (B_avg \geq B_hiwat \parallel W_avg \geq W_hiwat)$)時，切換至IFNET_MODEL_INPUT_POLL_ON，其中P_hiwat、B_hiwat及W_hiwat分別為封包、位元組及喚醒請求之高水位線(high-watermark)值。相反，當($P_avg \leq P_lowat \&& B_avg \leq B_lowat \&& W_avg \leq W_lowat$)時，切換至IFNET_MODEL_INPUT_POLL_OFF，其中



P_lowat、B_lowat及W_lowat為該等變數之低水位線值。

在一項實施例中，當前基於某些工作負載任意地選擇此等低及高水位線值，且應相應地調整該等低及高水位線值以適應未來工作負載(及變化之鏈路速度)。

在一項實施例中，大部分混合式輪詢邏輯駐存於dlil_rxpoll_input_thread_func()內，其中藉由基於以上邏輯呼叫驅動程式之input_ctl()回呼函式進行模式之間的轉變。關注速率以限制轉變，使得轉變並不過於頻繁地發生(在預設情況下，將保持時間設定為1秒)。

圖12說明在關閉/開啓輪詢時執行緒1201至1203之間的關係之一項實施例。

在一項實施例中，輪詢關閉/開啓模式之間的主要差異在於呼叫ifnet_input()或ifnet_input_extended()之內容脈絡及頻率。

在一項實施例中，當關閉輪詢時，將工作迴圈執行緒1201排程為主機CPU處置來自硬體之接收IRQ的部分；此IRQ用信號向主機通知裝置已將一或多個封包傳送至主機(例如，經由DMA)。無關於在硬體處進行之IRQ聯合的等級，藉由傳入封包之速率來驅控(drive)使此IOKit工作迴圈執行緒排程之頻率。與此排程(內容脈絡切換等)相關聯之成本相當顯著，尤其在給定如下事實時係如此：系統架構將所有IRQ投送至CPU0。因此，在一項實施例中，在偵測到高負載因數後即開啓輪詢。

當開啓輪詢時，借助於關閉接收IRQ而使工作迴圈執行緒1201靜止。封包仍自裝置傳送至主機，且封包累積於驅動程式1111之接收緩衝器中，直至封包藉由網路連接堆疊1108經由input_poll()回呼函式擷取為止。現處於作用中之輪詢者執行緒1202執行工作迴圈執行緒1201之等效功能性，惟排程此執行緒之頻率受網路連接堆疊1108嚴格控制

除外。

在一項實施例中，假定攤銷與自媒體接收封包相關之每封包處理成本，則輪詢導致效能改良。當開啓輪詢時，網路堆疊指導驅動程式進入輪詢模式。在處於輪詢模式中時，驅動程式將關閉其與封包自硬體到達之通知相關聯的接收中斷或設陷處置常式。封包將繼續前進至主機之記憶體(自裝置，經由DMA或等效物)，惟將不使CPU中斷除外。此情形減小CPU上之負載，此係因為每一中斷將正常地觸發一系列工作來對其進行處理；且該中斷將對效能具有一些負面影響，此係因為其優先於當時正在CPU上執行的事項。網路堆疊接著以1毫秒之時間間隔進行輪詢(在預設情況下；此為可組態的)，且在每一時間間隔期間自驅動程式拉取封包。若網路堆疊偵測到封包速率已下降，則退出輪詢模式，且重新啓用中斷。

在一項實施例中，輪詢可用以減小功率消耗(例如，當處於「低功率」模式時)，或輪詢係基於使用者活動(或不活動)。舉例而言，若系統處於低功率模式，則將此資訊供應至網路堆疊，且網路堆疊可接著選擇在所有適合網路介面上進入輪詢模式。將接著向網路堆疊告知系統何時不再處於低功率模式，使得可退出輪詢模式。

關於使用活動，若系統正忙於處置使用者介面輸入，則將此資訊供應至網路堆疊，且網路堆疊可接著選擇在所有適合網路介面上進入輪詢模式。將向網路堆疊告知系統何時不再忙於處置UI輸入，使得可退出輪詢模式。

接收佇列

在一項實施例中，ifnet之if_inp成員保持DLIL輸入執行緒1203之接收佇列。在一項實施例中，此資料結構含有說明於圖13中之資訊。

不同於其傳輸對應物，接收佇列與DLIL輸入執行緒執行個體而非與介面相關聯。如上文所提及，某些類型之介面共用共同(主)輸入

執行緒，而其他介面獲得其自身之專用輸入執行緒。亦不同於可存在多達N個傳輸佇列之傳輸情形，每輸入執行緒當前僅存在1個接收佇列執行個體。此結構亦保持關於以下各者之資訊：用於輸入之實際核心執行緒、工作迴圈1201，以及輪詢者執行緒1202。在一項實施例中，所有此等執行緒經組態以共用同一執行緒相似性標籤，以便使該等執行緒在同一處理器集合中進行排程(達成更好快取區域性)。在一項實施例中，機會性輪詢所需之參數(例如，模式、 $\{P,B,W\}_{avg}$ 、 $\{P,B,W\}_{\{lo,hi\}wat}$)亦駐存於此結構內。

鏈路事件

在一項實施例中，將與介面相關之事件自網路連接堆疊102發送至所連結排程器116及佇列管理邏輯115，且進一步發送至所有類別佇列執行個體110至112上。在需要時，此發送允許排程器、佇列管理邏輯115及其類別相應地調適其參數。該等事件如下：

事件	描述
CLASSQ_EV_LINK_SPEED	有效鏈路速率已改變，如藉由驅動程式報告(或歸因於TBR之改變)。
CLASSQ_EV_LINK_MTU	鏈路MTU已改變，如由SIOCSIFMTU所引起。
CLASSQ_EV_LINK_UP CLASSQ_EV_LINK_DOWN	鏈路狀態已改變(向上或向下)，如由設定/清除IFF_UP所引起。

如上文所提及，本發明之實施例包括對如下兩種不同排程模式之支援：(1)網路堆疊層處之排程，及(2)驅動程式層處之排程。驅動程式可選擇使用哪一類型之排程。在一項實施例中，若驅動程式正實施802.11n，則驅動程式可選擇使用藉由WMM定義之四個服務類別來執行排程(參見(例如)圖7，其說明服務類別與佇列執行個體之間的10:4映射)，而若驅動程式正實施任何其他介面(例如，102.11b/g、3G、乙太網路等)，則驅動程式可選擇在網路層處執行排程(參見(例如)圖6，其說明服務類別與佇列執行個體之間的1:1映射)。

在一項實施例中，在驅動程式管理之模型中，可如同網路堆疊管理之模型來設置所有佇列，但藉由驅動程式排程器160來執行排程。因而，基於驅動程式之排程器將接著基於優先權來針對特定類別請求用於每一移出佇列操作之數個封包(亦即，使用WMM之4個類別)。

雖然排程器116、160決定將封包自哪一佇列移出，但藉由佇列管理邏輯115實施之佇列處理演算法(亦稱為「丟棄程式(dropper)」，此係因為佇列處理演算法具有丟棄封包或將封包排入佇列之選項)判定在將佇列移出之前應將封包排入哪一佇列中。在一項實施例中，排程器116將封包交遞至佇列管理邏輯115，佇列管理邏輯115接著判定將封包丟棄抑或排入佇列。

如上文所提及，在一項實施例中，藉由網路堆疊使用之排程器116使用快速公平佇列處理(QFQ)，而驅動程式排程器160使用訊務類別佇列處理(TCQ)，其中之每一者已在上文進行了詳細描述。

流量控制及SFB佇列處理之額外細節

在一項實施例中，無關於選定排程器之類型而使用同一佇列管理邏輯115。對於驅動程式管理之排程器160及網路層級排程器116兩者，推測公平blue(SFB)可用作藉由佇列管理邏輯115實施之預設佇列處理演算法。如圖10A中所指示，流雜湊允許來自不同流/通訊端之封包易於在網路層及驅動程式層兩者內被追蹤到且由SFB演算法利用。當最初連接通訊端204時，計算雜湊值1001，將雜湊值1001儲存於通訊端資料結構中且使用歷時所連接通訊端204之使用期限。無論何時將資料自通訊端向下發送至較低層(例如，輸送層205、IP層206、sendq層306及驅動程式層150)，流雜湊值1001皆隨封包一起發送且用以獨特地識別通訊端/流。一旦識別到流，便在將每一新封包排入佇列時將在SFB分格1009(藉由圖10A中之變數C所指示)中之一或多者內

的計數器遞增，且在將每一封包移出佇列將計數器遞減。因此，每一流之當前經排入佇列之封包的數目為已知的，且用以執行對該流之流量控制。在一項實施例中，若特定流具有超出指定臨限值之經排入佇列的封包(如在圖10A中藉由計數器C₁>=FADV臨限值所指示)，則該流之機率值遞增某一區間。一旦機率達到其限值(例如，1)，則此情形指示存在過多封包(亦即，應用程式發送過快)，且丟棄此流之其他封包。

圖10A詳細地說明此機制。在一項實施例中，流雜湊用以對自通訊端204向下行進至較低層之每一封包加標籤，因此，此等較低層可辨識哪些封包屬於哪一流且將回饋提供至網路層。封包係排入與該流相關聯之sendq 306內，且如所提及，SFB使用流雜湊以針對每一流來將儲存於SFB分格1009內之計數器(C₁、C₂)遞增及遞減。SFB使用其記憶體資料結構中之計數器來判定是否已超出佇列處理臨限值。舉例而言，若流之經排入佇列之封包的數目為50(其處於臨限值)，則SFB儲存該流之狀態，且將回饋發送至輸送層，從而開啓對支援該流之通訊端204的流量控制。因此，流自正常狀態移至流量控制狀態且維持於此狀態下，同時封包自佇列移出且經由鏈路發送。

一旦佇列中之封包的數目降至臨限值以下，佇列處理演算法便喚醒流量警示執行緒1050，該流量警示執行緒1050喚醒通訊端204且關閉對通訊端204之流量控制。因此，通訊端可接著經由較低層隨意地向下傳輸封包，直至通訊端再次置於流量控制狀態為止。因此，圖10A說明如下機制：其提供回饋，且允許不依賴於TCP中諸如顯式壅塞通知(ECN)之內部機制而使同一主機上之流在流量控制狀態/正常狀態之間移動。此情形改良效能，此係因為減少了由於將資料向下發送至網路堆疊而僅被丟棄所進行之不必要工作。此外，該機制允許在不丟棄封包之情況下減小驅動程式中之佇列大小。此情形與如下較舊機

制形成對比：其中應用程式沿著堆疊盲目地傳輸封包而僅由驅動程式丟棄。對比而言，在圖10A中，因為併列監視，網路堆疊知曉何事項正在鏈路上進行，因此網路堆疊可使用此資訊來智慧地改變輸送機制之行為。

在圖10B中更詳細地說明之一項實施例中，SFB演算法獲取封包之雜湊值且計算另一雜湊，該另一雜湊對於SFB而言為獨特的且用以捨取該流之位元。如所說明，藉由通訊端計算之原始雜湊值為4個位元組(01、02、03、04)，可將此雜湊值提供至SFB併列處理邏輯1060之雜湊產生器1065。某些流雜湊可具備值0，且用以識別未對之使用此處所描述流量控制機制的流。在一項實施例中，SFB產生隨機數，且使用隨機數(A、B、C、D)來計算新雜湊。SFB使用此等四個值來填入如下兩個分格集合：當前集合及陰影複製集合(shadow set)，該等集合中之每一者具有兩個層級，如圖10B中所展示。將兩個層級實施為陣列，該陣列使用新雜湊值來標索引(在一項實施例中，每一槽為256個位元)。因此，使用流雜湊值將流映射至SFB分格中。為了檢查流是否存在，可檢查SFB分格。SFB演算法之布隆過濾器功能能夠檢查特定流是否不存在於SFB分格中。若存在一個流，則可對該流標索引以判定與該流相關聯之資料(例如，上文所提及之機率及計數器)。只要流仍存在，便藉由SFB來維持此資料。使用陰影複製集合，此係因為SFB以某一時間間隔進行再雜湊。因為布隆過濾器之性質，許多事項可映射至同一分格。若存在許多通訊端，則正發送過多資料之些通訊端可影響映射至同一分格之並非正發送許多資料的其他通訊端。因此，SFB基於新隨機數週期性地重新計算內部雜湊。在再雜湊之後，可針對每一流將分格來回移動。SFB演算法之額外細節可見於參考文獻**Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness**中，該參考案作為附錄B附加至本專利申請案且以

引用之方式併入本文中。在此狀況下，陰影複製集合可用以將資料移動至新分格。

應用程式驅動式訊務分類及流量回饋，包括機會性行爲、流量控制、訊務暫時中止及輸送層改良

上文關於圖6至圖8描述之應用程式驅動式訊務分類可用以啓用各種機會性行爲。如所提及，每一應用程式可顯式地規定應將何訊務類別用於其通訊端。舉例而言，據稱一個應用程式正使用諸如BK_SYS之低優先權訊務類別備份資料，但使用者想要進行語音通話。因為GSM工作之方式，若語音通話正在進行中同時正發送網路資料，則此情形使語音通話惡化或中斷之機率增加。類似地，若使用者在通話的同時開啓web瀏覽器，則該操作亦可影響通話。為了減輕此影響，在一項實施例中，當使用者進行語音通話時，訊務類別用以抑制不必要訊務。可在語音通話期間暫時中止任何非緊急訊務(例如，背景訊務BKS_SYS)。在一項實施例中，對於暫時中止之通訊端，當移出佇列發生時，系統假稱不存在要移出佇列之封包，且在排入佇列發生時，丟棄封包。因此，傳輸應用程式將此情形視為如同網路已停止一般，且可在再次嘗試之前等待。若呼叫相對快速，則傳輸應用程式在呼叫已結束時將重新繼續(不完全斷開通訊端)。此等技術可用於不同於BK_SYS之訊務類別(例如，BK、BE等)。在一項實施例中，上文所描述之流量控制/警示機制亦用以在此等情形下暫時中止通訊端(例如，若暫時中止，則發送流量控制指示)。因此，藉由對某些通訊端之訊務進行分類(例如，BK_SYS)，提供用於處置訊務暫時中止之智慧回饋機制，且在核心中啓用此等通訊端之機會性行爲。

請注意，暫時中止狀態不同於流量控制狀態。在流量控制狀態下，應用程式可仍發送資料，但在暫時中止狀態下，阻斷鏈路(例如，因為語音通話花費長時間)。因此，在暫時中止狀態下，停止發

送更多封包為有益的，此係因為該等封包將僅被丟棄(因為將併列暫時中止)。在一項實施例中，暫時中止之應用程式可選擇啓動計時器，且若暫時中止過長，則簡單地將連接關閉。

簡言之，在一項實施例中，當接收到/進行語音通話時，執行以下步驟：

(1)系統中之授權實體將蜂巢式網路介面組態為機會性節流模式。

(2)在每一受影響介面之封包排程器上組態機會性模式。

(3)在此機會性模式中，排程器遍歷待暫時中止之所有傳輸併列；目前，此情形僅應用於BK_SYS傳輸併列。每一受影響傳輸併列進入暫時中止模式；清除所有現有封包，且其他排入併列將引起丟棄。

(4)所有機會性通訊端(具有BK_SYS類別之通訊端)將接收「暫時中止事件」。

在一項實施例中，當終止語音通話時，執行以下步驟：

(1)系統中之授權實體自蜂巢式網路介面移除機會性節流模式。

(2)每一受影響介面之封包排程器退出機會性節流模式。

(3)排程器遍歷在此機會性模式中暫時中止之所有傳輸併列；目前，此情形僅應用於BK_SYS傳輸併列。使每一受影響傳輸併列重新繼續；允許其他排入併列。

(4)所有機會性通訊端(具有BK_SYS類別之通訊端)將接收「重新繼續事件」。

此外，在接收側實現效能最佳化，本文中稱為「大接收卸載」。在一項實施例中，此效能最佳化藉由以下操作來實現：藉由呼叫網路堆疊中之函式來減小每封包之成本。舉例而言，在接收到10個封包時，可僅處理1或2個封包而非處置所有10個封包。對於諸如AV類別

(串流視訊)之某些類別，可啓用此最佳化。藉由AV視訊應用程式，軟體在視訊開始播放之前對視訊進行緩衝歷時數秒。因此，因為此情形，此應用程式可使用大接收超載技術來得到效能益處，此係因為該等技術對延遲不敏感。

在一項實施例中，因為沿著堆疊向上提供回饋，所以可針對某些通訊端調整TCP之發送及接收方式。舉例而言，若應用程式將流分類為BK或BK_SYS，則TCP將較不積極。舉例而言，若通訊端為BK且偵測到此鏈路上之高壅塞，則應用程式可後移且得到對其他流之更好回應時間(例如，可使備份至網路之通訊端延遲而在另一通訊端上接收HTTP)。有可能實現全部此情形，此係因為應用程式能夠顯式地指定通訊端訊務類別。

如先前所提及，在一項實施例中，最高種類(CTL)用於網路控制封包，該等網路控制封包可包括ARP、ACK回應、IPV6之相鄰者發現、多播加入及離開、DCP相關封包、IPV6站路器廣告封包，及DNS封包。因此，若使用者正使用較低優先權通訊端執行大量上載，則DNS操作將不會受到延遲。類似地，由於較低優先權通訊端上之活動，ACK回應將不會受到延遲。此類型之控制封包極小，且可在無任何延遲情況下傳出。

此外，如所提及，上文所描述之技術及架構在流量控制及暫時中止狀態期間提供內建式TCP及UDP退讓機制。通常，TCP自身藉由退讓(例如，使用如上文所描述之ECN)來對壅塞作出回應。TCP發送封包，且若並未獲得確認(ACK)，則TCP再次發送封包且隨著每一重傳輸而退讓。若已知封包仍處於介面佇列151至153中且將僅在鏈路變為可用之後發送，則此重傳輸浪費資源。因此，在此情形下，不需要退讓且使用內建式TCP或UDP機制來重傳輸。實情為，使用本文中所描述之技術，可發出流量控制警示以停用重傳輸功能。

此外，在一項實施例中，可微調用於TCP之重傳輸計時器以更高效地操作。通常，當未接收到ACK時，TCP使重傳輸計時器後移(例如，每次增加為2倍)。因此，重傳輸計時器可變成若干秒。由於如上文所描述提供來自介面之回饋，因此此無需發生此後移。一旦獲得流量控制警示，便可自動等待，此係因為知曉封包經排入佇列(且並不需要保持重發送)。然而，在一項實施例中，可在已關閉對通訊端之流量控制之後使用重傳輸計時器。

此外，始終將流量警示傳播至不將更多資料寫入至TCP之應用程式。應用程式可因此丟棄所有過時資料(例如，舊視訊圖框)，且在流量控制關閉時發送較新資料。此情形對於互動式視訊資料特別有益，其中若應用程式不丟棄過時資料，則音訊及視訊可脫離同步。

類似原理可應用於藉由UDP傳輸之應用程式。在無本文中所描述之流量警示技術的情況下，UDP不將任何回饋提供至其應用程式。因此，若介面中斷，則應用程式不知曉且繼續傳輸封包，封包因此必須在堆疊之較低層級處進行緩衝，從而浪費資源。相比而言，當處於流量控制狀態下時，可立即中斷應用程式進行之所有寫入(藉此節省緩衝資源)。進行以上操作會顯著減小由對封包進行緩衝所產生之延遲。

用於處置傳入網路訊務、改良網路效能之處置阻斷服務攻擊(denial-of-service attack)的機會性輪詢機制

如上文所提及，在一項實施例中，具有RXPOLL能力之介面在輸入輪詢關閉操作模式與輪詢開啓(或舊版)操作模式(例如，IFNET_MODEL_INPUT_POLL_OFF 與 IFNET_MODEL_INPUT_POLL_ON)之間轉變。在一項實施例中，將輸入輪詢關閉模式用作預設/初始模式，且在網路堆疊判定負載因數為低時由網路堆疊選擇該模式。可藉由評估變數P_avg、B_avg及



W_avg(分別為封包、位元組及喚醒請求之數目的值)來判定負載因數。參考圖12中之DLL輸入執行緒1203，在一項實施例中，當($P_{avg} \geq P_{hiwat} \&& (B_{avg} \geq B_{hiwat} || W_{avg} \geq W_{hiwat})$)時，切換至輸入輪詢開啓模式，其中 P_{hiwat} 、 B_{hiwat} 及 W_{hiwat} 為變數之低水位線值。

相反，當($P_{avg} \leq P_{lowat} \&& B_{avg} \leq B_{lowat} \&& W_{avg} \leq W_{lowat}$)時，切換至輪詢關閉模式，其中 P_{lowat} 、 B_{lowat} 及 W_{lowat} 為變數之低水位線值。

在一項實施例中，可基於某些工作負載任意地選擇此等低及高水位線值，且應相應地調整該等低及高水位線值以適應未來工作負載(及變化之鏈路速度)。

以此方式基於高負載因數使輪詢開啓，此改良效能且防止阻斷服務攻擊。此係因為，藉由使用輪詢機制，網路層處之接收佇列將僅在其具有足以將驅動程式層處之佇列之封包排入佇列的空間時才向該等佇列請求該等封包。當該接收佇列不具有空間時，將不對請求更多封包進行輪詢且將藉由介面丟棄封包(亦即，當驅動程式層佇列填滿時)。因此，在驅動程式層處防止了阻斷服務，且不會將其向上傳播至網路層。

不同API實施例

在一項實施例中實施之API為藉由軟體組件(下文中為「API實施軟體組件」)實施之介面，該介面允許不同軟體組件(下文中為「API呼叫軟體組件」)存取及使用一或多個函式、方法、程序、資料結構及/或藉由API實施軟體組件提供之其他服務。舉例而言，API允許API呼叫軟體組件之開發者(其可為第三方開發者)充分利用藉由API實施軟體組件提供之指定特徵。可存在一個API呼叫軟體組件，或可存在一個以上此種軟體組件。API可為原始程式碼介面，電腦系統或程式

庫提供該原始程式碼介面以便支援對來自軟體應用程式之服務的請求。可依據程式設計語言而非關於資料佈置於記憶體中之方式的顯式低階描述來指定API，可在組建應用程式時解譯或編譯該程式設計語言。

API定義API呼叫軟體組件在存取及使用API實施軟體組件之指定特徵時使用的語言及參數。舉例而言，API呼叫軟體組件經由一或多個API呼叫(有時稱為函式或方法呼叫)來存取API實施軟體組件之指定特徵，該一或多個API呼叫藉由API公開(expose)。API實施軟體組件可回應於來自API呼叫軟體組件之API呼叫而經由API將值傳回。雖然API定義API呼叫之語法及結果(例如，調用API呼叫之方式及API呼叫進行之操作)，但API通常不揭露API呼叫實現由API呼叫所指定之函式的方式。經由一或多個應用程式設計來在介面呼叫軟體(API呼叫軟體組件)與API實施軟體組件之間傳送各種函式呼叫或訊息。傳送函式呼叫或訊息可包括發出、起始、調用、呼叫、接收、傳回功能呼叫或訊息或對功能呼叫或訊息作出回應。因此，API呼叫軟體組件可傳送呼叫，且API實施軟體組件可傳送呼叫。

藉由實例，API實施軟體組件2010及API呼叫軟體組件可為作業系統、程式庫、裝置驅動程式、API、應用程式，或其他軟體模組(應理解，API實施軟體組件及API呼叫軟體組件彼此可為相同或不同類型之軟體模組)。API呼叫軟體組件可為區域軟體組件(亦即，在與API實施軟體組件相同之資料處理系統上)或遠端軟體組件(亦即，在與API實施軟體組件不同之資料處理系統上)，該遠端軟體組件經由網路借助於API與API實施軟體組件通信。應理解，API實施軟體組件亦可充當API呼叫軟體組件(亦即，其可對藉由不同API實施軟體組件公開之API進行API呼叫)，且API呼叫軟體組件亦可藉由實施向不同API呼叫軟體組件公開之API而充當API實施軟體組件。

API可允許用不同程式設計語言撰寫之多個API呼叫軟體組件與API實施軟體組件通信(因此，API可包括用於轉譯API實施軟體組件與API呼叫軟體組件之間的呼叫及傳回項之特徵)；然而，可依據特定程式設計語言來實施API。

圖14說明API架構之一項實施例，該API架構包括實施API 1420之API實施軟體組件1410(例如，作業系統、程式庫、裝置驅動程式、API、應用程式或其他軟體模組)。API 1420指定API實施軟體組件之可由API呼叫軟體組件1430使用的一或多個函式、方法、類別、物件、協定、資料結構、格式及/或其他特徵。API 1420可指定至少一個呼叫慣例，該至少一個呼叫慣例指定以下各者：API實施軟體組件中之函式自API呼叫軟體組件接收參數的方式，及該函式將結果傳回至API呼叫軟體組件之方式。API呼叫軟體組件1430(例如，作業系統、程式庫、裝置驅動程式、API、應用程式或其他軟體模組)經由API 1420進行API呼叫以存取及使用API實施軟體組件1410之藉由API 1420指定的特徵。API實施軟體組件1410可回應於API呼叫而經由API 1420將值傳回至API呼叫軟體組件1430。

應瞭解，API實施軟體組件1410可包括額外函式、方法、類別、資料結構及/或未經由API 1420指定且不可用於API呼叫軟體組件1430之其他特徵。應理解，API呼叫軟體組件1430可在與API實施軟體組件1410相同之系統上，或可位於遠端且經由網路使用API 1420來存取API實施軟體組件1410。雖然圖14說明與API 1420互動之單一API呼叫軟體組件1430，但應理解，可用不同於API呼叫軟體組件1430之語言(或同一語言)撰寫的其他API呼叫軟體組件可使用API 1420。

API實施軟體組件1410、API 1420及API呼叫軟體組件1430可儲存於機器可讀媒體中，該機器可讀媒體包括用於以機器(例如，電腦或其他資料處理系統)可讀形式儲存資訊之任何機制。舉例而言，機

器可讀媒體包括磁碟、光碟、隨機存取記憶體、唯讀記憶體、快閃記憶體裝置等。

在圖15(「軟體堆疊」)之例示性實施例中，應用程式可使用若干服務API進行對服務1或2之呼叫且使用若干OS API進行對作業系統(OS)之呼叫。服務1及2可使用若干OS API對OS進行呼叫。

請注意，服務2具有兩個API，其中之一者(服務2之API 1)接收來自應用程式1之呼叫且將值傳回至應用程式1，且另一者(服務2之API 2)接收來自應用程式2之呼叫且將值傳回至應用程式2。服務1(例如，其可為軟體程式庫)進行對OS API 1之呼叫且接收自OS API 1傳回之值，且服務2(例如，其可為軟體程式庫)進行對OS API 1及OS API 2兩者之呼叫且接收自OS API 1及OS API 2兩者傳回之值。應用程式2進行對OS API 2之呼叫且接收自OS API 2傳回之值。

例示性資料處理裝置及介面

圖16為說明可用於本發明之些實施例中之例示性電腦系統的方塊圖。應理解，雖然圖16說明電腦系統之各種組件，但圖16並非意欲表示將組件互連之任何特定架構或方式，此係因為此等細節與本發明並無密切關係。應瞭解，具有更少組件或更多組件之其他電腦系統亦可供本發明使用。

如圖16中所說明，呈電腦系統2300(其為資料處理系統之一種形式)包括與以下各者耦接之匯流排2350：處理系統2320、電源供應器2325、記憶體2330，及非揮發性記憶體2340(例如，硬碟機、快閃記憶體、相變記憶體(PCM)等)。匯流排2350彼此可經由如此項技術中熟知之各種橋接器、控制器及/或配接器而連接。處理系統2320可自記憶體2330及/或非揮發性記憶體2340擷取指令，且執行指令以執行如上文所描述之操作。匯流排2350將以上組件互連在一起且亦將彼等組件互連至選用之銜接台2360、顯示控制器與顯示裝置2370、輸入/輸

出裝置2380(例如，NIC(網路介面卡)、游標控制件(例如，滑鼠、觸控式螢幕、觸控墊等)、鍵盤等)及選用之無線收發器2390(例如，藍芽、WiFi、紅外線等)。

圖17為說明可用於本發明之些實施例中之例示性資料處理系統的方塊圖。舉例而言，資料處理系統2400可為手持型電腦、個人數位助理(PDA)、行動電話、攜帶型遊戲系統、攜帶型媒體播放器，可包括行動電話、媒體播放器及/或遊戲系統之平板或手持型計算裝置。作為另一實例，資料處理系統2400可為網路電腦，或另一裝置內之嵌入式處理裝置。

根據本發明之一項實施例，資料處理系統2400之例示性架構可用於上文所描述之行動裝置。資料處理系統2400包括處理系統2420，該處理系統2420可包括積體電路上之一或多個微處理器及/或一系統。處理系統2420與以下各者耦接：記憶體2410、電源供應器2425(其包括一或多個電池)、音訊輸入/輸出件2440、顯示控制器及顯示裝置2460、選用之輸入/輸出件2450、輸入裝置2470，及無線收發器2430。應瞭解，在本發明之某些實施例中，圖24中未展示之額外組件亦可為資料處理系統2400之部分，且在本發明之某些實施例中，可使用比圖16中展示之組件少的組件。此外，應瞭解，圖16中未展示之一或多個匯流排可用以將如此項技術中熟知之各種組件互連。

記憶體2410可儲存資料及/或程式以供資料處理系統2400執行。音訊輸入/輸出件2440可包括麥克風及/或揚聲器，以(例如)經由揚聲器及麥克風播放音樂及/或提供電話功能性。顯示控制器及顯示裝置2460可包括圖形使用者介面(GUI)。無線(例如，RF)收發器2430(例如，WiFi收發器、紅外線收發器、藍芽收發器、無線蜂巢式電話收發器等)可用以與其他資料處理系統通信。一或多個輸入裝置2470允許使用者將輸入提供至系統。此等輸入裝置可為小鍵盤、鍵盤、觸控面

板、多點觸控面板等。選用之其他輸入/輸出件2450可為銜接站之連接器。

本發明之實施例可包括如上文所闡述之各種步驟。該等步驟可以使通用或專用處理器執行某些步驟之機器可執行指令體現。或者，可藉由特定硬體組件(其含有用於執行該等步驟之固線式邏輯)或藉由經程式化之電腦組件及定製硬體組件的任何組合來執行此等步驟。

亦可將本發明之元件提供為用於儲存機器可執行程式碼之機器可讀媒體。機器可讀媒體可包括(但不限於)軟碟、光碟、CD-ROM及磁光碟、ROM、RAM、EPROM、EEPROM、磁卡或光卡，或適合於儲存電子程式碼之其他類型的媒體/機器可讀媒體。

貫穿前述描述，為達成解釋之目的，闡述眾多特定細節以便提供對本發明之透徹理解。然而，熟習此項技術者將易於顯而易見，可在無此等特定細節中之一些之情況下實踐本發明。舉例而言，熟習此項技術者將易於顯而易見，可將本文中所描述之功能模組及方法實施為軟體、硬體或其任何組合。此外，雖然本文中在行動計算環境(亦即，使用行動裝置120至123；601至603)之情境內描述本發明之實施例，但本發明之基本原理並不限於行動計算實施。實際上，可在一些實施例中使用任何類型之客戶端或同級資料處理裝置，包括(例如)桌上型電腦或工作站電腦。因此，應依據下文之申請專利範圍來判斷本發明之範疇及精神。

【符號說明】

- | | |
|-----|---------|
| 101 | 客戶端計算裝置 |
| 102 | 網路連接堆疊層 |
| 105 | 應用程式 |
| 106 | 應用程式 |
| 107 | 應用程式 |



- 108 應用程式設計介面(API)
- 110 傳輸佇列/佇列執行個體/網路連接佇列
- 111 傳輸佇列/佇列執行個體/網路連接佇列
- 112 傳輸佇列/佇列執行個體/網路連接佇列
- 113 佇列
- 115 佇列管理邏輯
- 116 封包排程器模組/網路層級排程器
- 120 伺服器
- 121 客戶端
- 122 客戶端裝置/伺服器
- 123 伺服器
- 150 網路/驅動程式層
- 151 驅動程式佇列/介面佇列
- 152 驅動程式佇列/介面佇列
- 153 驅動程式佇列/介面佇列
- 155 驅動程式佇列管理邏輯
- 160 驅動程式排程器
- 171 連續流量控制回饋/回饋信號
- 172 連續回饋
- 201 應用程式
- 202 封包分類器
- 203 應用程式設計介面(API)
- 204 通訊端層
- 205 輸送層
- 206 網際網路封包層
- 207 流量警示模組

208	資料鏈路介面層(DLIL)/網路連接應用程式設計介面 (API)
210	類別佇列/類別佇列執行個體
211	核心程式設計介面(KPI)
301	應用程式
302	網路堆疊
303	IO網路連接介面
304	IO輸出佇列
305	驅動程式
306	發送佇列
307	驅動程式
402	啓動者執行緒
601	封包服務類別
602	封包服務類別
603	封包服務類別
604	封包服務類別
605	封包服務類別
606	封包服務類別
607	封包服務類別
608	封包服務類別
609	封包服務類別
610	封包服務類別
611	封包佇列執行個體
612	封包佇列執行個體
613	封包佇列執行個體
614	封包佇列執行個體

615	封包佇列執行個體
616	封包佇列執行個體
617	封包佇列執行個體
618	封包佇列執行個體
619	封包佇列執行個體
620	封包佇列執行個體
630	群組
640	群組
650	群組
660	群組
701	佇列執行個體
702	佇列執行個體
703	佇列執行個體
704	佇列執行個體
1001	流雜湊值
1009	SFB分格
1050	流量警示執行緒
1060	SFB佇列處理邏輯
1065	雜湊產生器
1101	請求應用程式
1102	網路堆疊
1103	接收佇列執行個體
1104	IO網路連接介面
1105	驅動程式
1107	請求應用程式
1108	網路連接堆疊

- 1109 接收併列執行個體
- 1110 IP網路連接介面
- 1111 驅動程式
- 1112 硬體
- 1201 工作迴圈執行緒
- 1202 輪詢者執行緒
- 1203 DLL輸入執行緒
- 1410 API實施軟體組件
- 1420 應用程式設計介面(API)
- 1430 API呼叫軟體組件
- 2300 電腦系統
- 2320 處理系統
- 2325 電源供應器
- 2330 記憶體
- 2340 非揮發性記憶體
- 2350 汱流排
- 2360 選用之銜接站
- 2370 顯示控制器與顯示裝置
- 2380 輸入/輸出裝置
- 2390 選用之無線收發器
- 2400 資料處理系統
- 2410 記憶體
- 2420 處理系統
- 2425 電源供應器
- 2430 無線收發器
- 2440 音訊輸入/輸出件

- 2450 選用之輸入/輸出件
- 2460 顯示控制器及顯示裝置
- 2470 輸入裝置

申請專利範圍

1. 一種用於在一客戶端裝置上管理封包排程之電腦實施方法，該方法包含：
 - 接收待傳輸之一封包；
 - 在一網路堆疊層級處將該封包排入一佇列中；
 - 判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；
 - 若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及
 - 若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。
2. 如請求項1之方法，其中針對某些Wi-Fi網路介面，封包排程係在該驅動程式層級處執行，但針對所有其他媒體存取控制(MAC)介面，封包排程係在該網路連接堆疊層級處執行。
3. 如請求項2之方法，其中該等其他MAC介面包括乙太網路介面及無線蜂巢式介面。
4. 如請求項2之方法，其中針對802.11n資料訊務，封包排程係在該驅動程式層級處執行，但針對802.11a、802.11b或802.11g資料訊務，封包排程係在該網路層處執行。
5. 如請求項4之方法，其中802.11n驅動程式實施無線多媒體延伸(WME)以根據四個優先權等級來對網路訊務進行排程，該四個優先權等級為語音、視訊、最佳努力及背景。
6. 如請求項1之方法，其進一步包含：
 - 根據一特定服務類別對該封包進行分類；及
 - 基於該封包藉以分類之該服務類別將該封包排入一特定傳輸

佇列執行個體中。

7. 如請求項6之方法，其中網路連接堆疊將一封包排入佇列以供傳輸，且向該驅動程式層級通知該封包何時準備好傳輸。
8. 如請求項7之方法，其進一步包含：

 將一回饋信號自該驅動程式層級提供至該網路連接堆疊層級，以確保該網路連接堆疊知曉藉由驅動程式管理之一通信鏈路的網路連接條件。

9. 如請求項8之方法，其中該網路層級基於與該通信鏈路相關聯之該等偵測到的網路連接條件執行封包排程。
10. 如請求項1之方法，其進一步包含：

 將一介面連結至該網路連接堆疊；及

 回應性地宣告一介面類型及關於以下情形之一指示：針對該介面，封包排程將在該網路堆疊層級處抑或該驅動程式層級處執行。

11. 如請求項10之方法，其中該介面在以下狀態之間切換：在該網路堆疊層級處進行封包排程；及在該驅動程式層級處進行封包排程。
12. 如請求項10之方法，其中基於該介面類型及排程模型，該網路堆疊選擇最適當封包排程器及佇列處理演算法。
13. 如請求項6之方法，其中該網路堆疊自動地判定與該傳輸佇列相關聯之佇列參數，該等佇列參數包括佇列大小、流量控制水位線及丟棄臨限值。
14. 如請求項6之方法，其中該網路堆疊層級自動地組態封包排程參數，該等封包排程參數包括跨越佇列之鏈路共用分佈。
15. 如請求項1之方法，其中該等接收、判定及排程操作係在一終端使用者不介入之情況下自動地執行。



16. 一種上面儲存有程式碼之機器可讀媒體，該程式碼在由一客戶端裝置執行時使該客戶端裝置執行以下操作：
 - 接收待傳輸之一封包；
 - 在一網路堆疊層級處將該封包排入一佇列中；
 - 判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；
 - 若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及
 - 若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。
17. 如請求項16之機器可讀媒體，其中針對某些Wi-Fi網路介面，封包排程係在該驅動程式層級處執行，但針對所有其他媒體存取控制(MAC)介面，封包排程係在該網路連接堆疊層級處執行。
18. 如請求項17之機器可讀媒體，其中該等其他MAC介面包括乙太網路介面及無線蜂巢式介面。
19. 如請求項17之機器可讀媒體，其中針對802.11n資料訊務，封包排程係在該驅動程式層級處執行，但針對802.11a、802.11b或802.11g資料訊務，封包排程係在該網路層處執行。
20. 如請求項19之機器可讀媒體，其中802.11n驅動程式實施無線多媒體延伸(WME)以根據四個優先權等級來對網路訊務進行排程，該四個優先權等級為語音、視訊、最佳努力及背景。
21. 如請求項15之機器可讀媒體，其進一步包含使該客戶端裝置執行以下操作之額外程式碼：
 - 根據一特定服務類別對該封包進行分類；及
 - 基於該封包藉以分類之該服務類別將該封包排入一特定傳輸佇列執行個體中。

22. 如請求項21之機器可讀媒體，其中當封包排程當前係在該網路連接堆疊層級處執行時，網路連接堆疊將一封包排入佇列以供傳輸，且向該驅動程式層級通知該封包何時準備好傳輸。
23. 如請求項22之機器可讀媒體，其進一步包含使該客戶端裝置執行以下操作之額外程式碼：

 將一回饋信號自該驅動程式層級提供至該網路連接堆疊層級，以確保該網路連接堆疊知曉藉由驅動程式管理之一通信鍊路的網路連接條件。

24. 如請求項23之機器可讀媒體，其中該網路層級基於與該通信鍊路相關聯之該等偵測到的網路連接條件執行封包排程。
25. 如請求項16之機器可讀媒體，其進一步包含：

 將一介面連結至該網路連接堆疊；及

 回應性地宣告一介面類型及關於以下情形之一指示：針對該介面，封包排程將在該網路堆疊層級處抑或該驅動程式層級處執行。

26. 如請求項25之機器可讀媒體，其中該介面在以下狀態之間切換：在該網路堆疊層級處進行封包排程；及在該驅動程式層級處進行封包排程。
27. 如請求項25之機器可讀媒體，其中基於該介面類型及排程模型，該網路堆疊選擇最適當封包排程器及佇列處理演算法。
28. 如請求項21之機器可讀媒體，其中該網路堆疊自動地判定與該傳輸佇列相關聯之佇列參數，該等佇列參數包括佇列大小、流量控制水位線及丟棄臨限值。
29. 如請求項21之機器可讀媒體，其中該網路堆疊層級自動地組態封包排程參數，該等封包排程參數包括跨越佇列之鏈路共用分佈。

30. 如請求項16之機器可讀媒體，其中該等接收、判定及排程操作係在一終端使用者不介入之情況下自動地執行。
31. 一種客戶端裝置，其具有用於儲存程式碼之一記憶體及用於處理該程式碼以執行以下操作的一處理器：
 - 接收待傳輸之一封包；
 - 在一網路堆疊層級處將該封包排入一佇列中；
 - 判定當前正在一驅動程式層級處抑或在一網路連接堆疊層級處執行封包排程；
 - 若當前正在該網路堆疊層級處執行排程，則在該網路堆疊層級處自該佇列選擇該封包以供傳輸；及
 - 若當前正在該驅動程式層級處執行排程，則在該驅動程式層級處自該佇列選擇該封包以供傳輸。
32. 如請求項31之客戶端裝置，其中針對某些Wi-Fi網路介面，封包排程係在該驅動程式層級處執行，但針對所有其他媒體存取控制(MAC)介面，封包排程係在該網路連接堆疊層級處執行。
33. 如請求項32之客戶端裝置，其中該等其他MAC介面包括乙太網路介面及無線蜂巢式介面。
34. 如請求項32之客戶端裝置，其中針對802.11n資料訊務，封包排程係在該驅動程式層級處執行，但針對802.11a、802.11b或802.11g資料訊務，封包排程係在該網路層處執行。
35. 如請求項34之客戶端裝置，其中802.11n驅動程式實施無線多媒體延伸(WME)以根據四個優先權等級來對網路訊務進行排程，該四個優先權等級為語音、視訊、最佳努力及背景。
36. 如請求項31之客戶端裝置，其進一步包含使該客戶端裝置執行以下操作之額外程式碼：
 - 根據一特定服務類別對該封包進行分類；及

基於該封包藉以分類之該服務類別將該封包排入一特定傳輸佇列執行個體中。

37. 如請求項36之客戶端裝置，其中網路連接堆疊將一封包排入佇列以供傳輸，且向該驅動程式層級通知該封包何時準備好傳輸。
38. 如請求項37之客戶端裝置，其進一步包含使該客戶端裝置執行以下操作之額外程式碼：

將一回饋信號自該驅動程式層級提供至該網路連接堆疊層級，以確保該網路連接堆疊知曉藉由驅動程式管理之一通信鏈路的網路連接條件。

39. 如請求項38之客戶端裝置，其中該網路層級基於與該通信鏈路相關聯之該等偵測到的網路連接條件執行封包排程。
40. 如請求項28之客戶端裝置，其進一步包含：

將一介面連結至該網路連接堆疊；及

回應性地宣告一介面類型及關於以下情形之一指示：針對該介面，封包排程將在該網路堆疊層級處抑或該驅動程式層級處執行。

41. 如請求項37之客戶端裝置，其中該介面在以下狀態之間切換：在該網路堆疊層級處進行封包排程；及在該驅動程式層級處進行封包排程。
42. 如請求項37之客戶端裝置，其中基於該介面類型及排程模型，該網路堆疊選擇最適當封包排程器及佇列處理演算法。
43. 如請求項33之客戶端裝置，其中該網路堆疊自動地判定與該傳輸佇列相關聯之佇列參數，該等佇列參數包括佇列大小、流量控制水位線及丟棄臨限值。
44. 如請求項33之客戶端裝置，其中該網路堆疊層級自動地封包排

程參數，該等封包排程參數包括跨越佇列之鏈路共用分佈。

45. 如請求項40之客戶端裝置，其中該等接收、判定及排程操作係在一終端使用者不介入之情況下自動地執行。

圖 1A

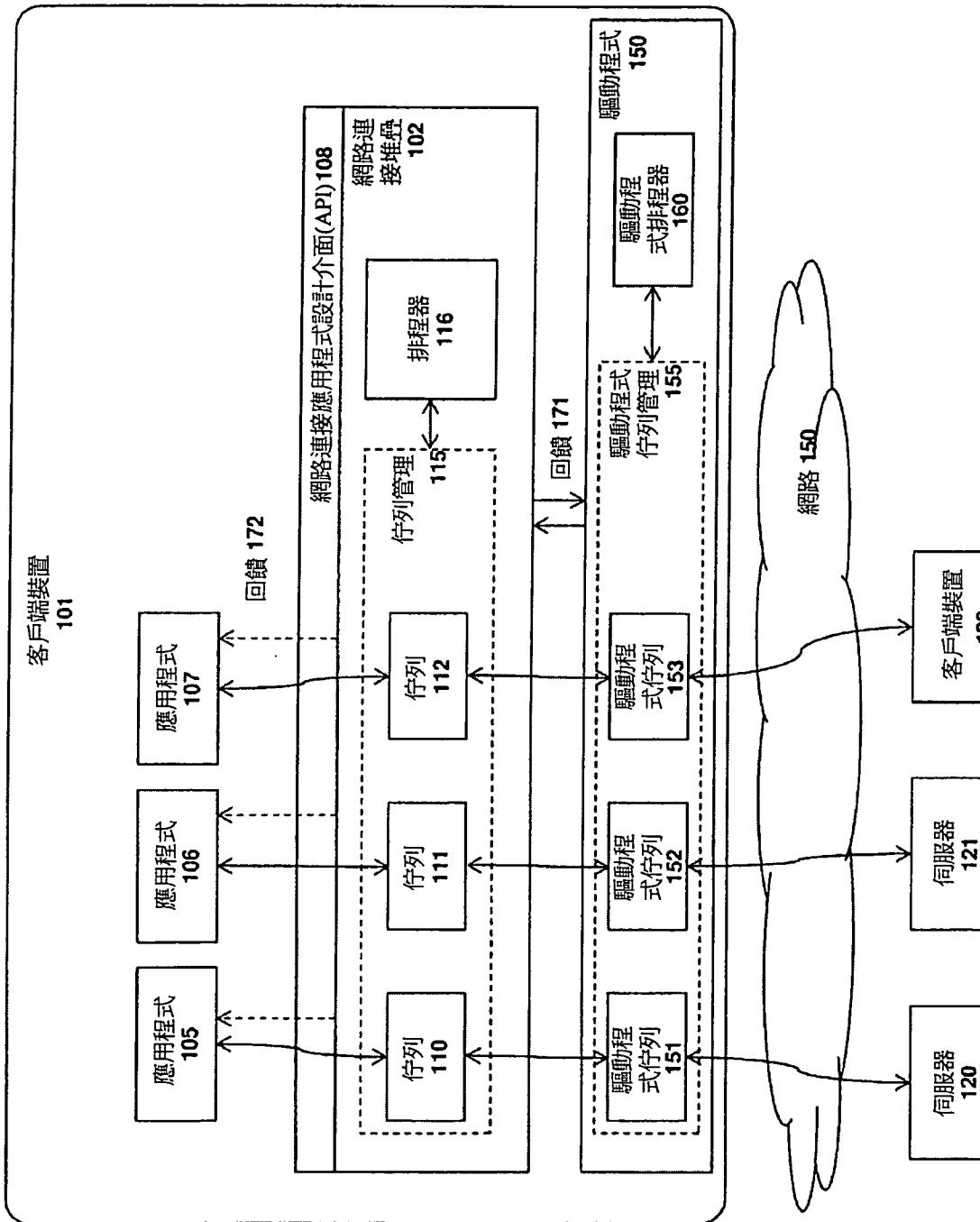


圖1A

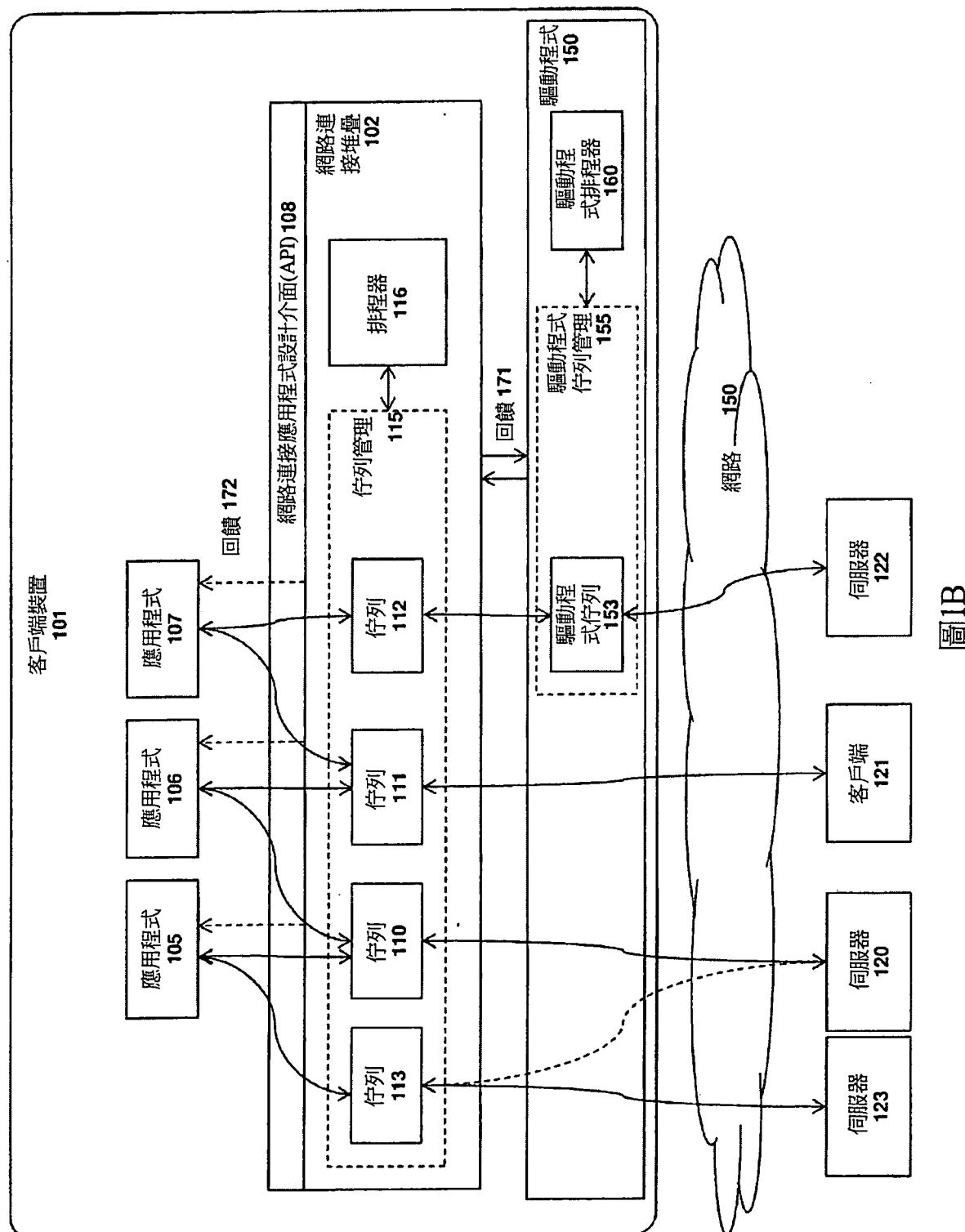
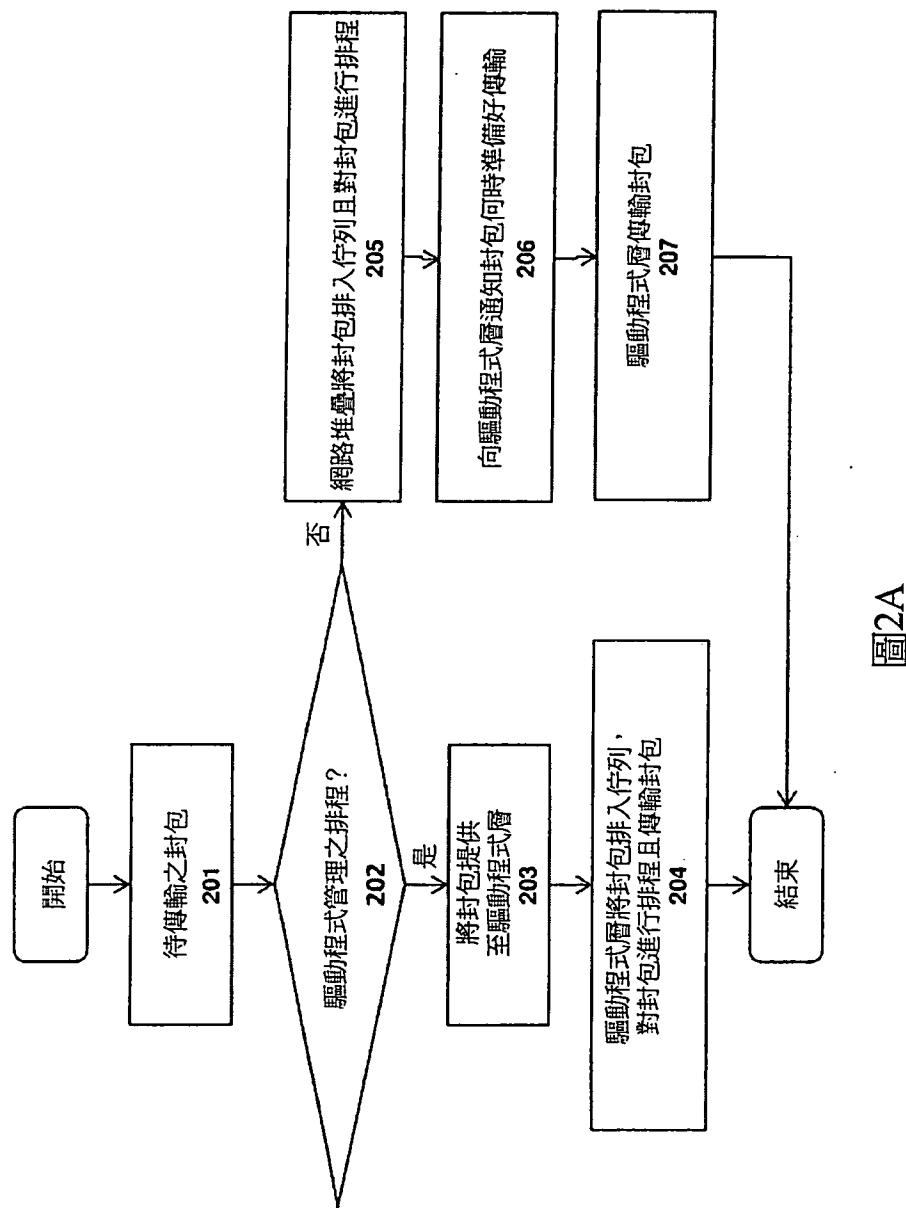


圖1B



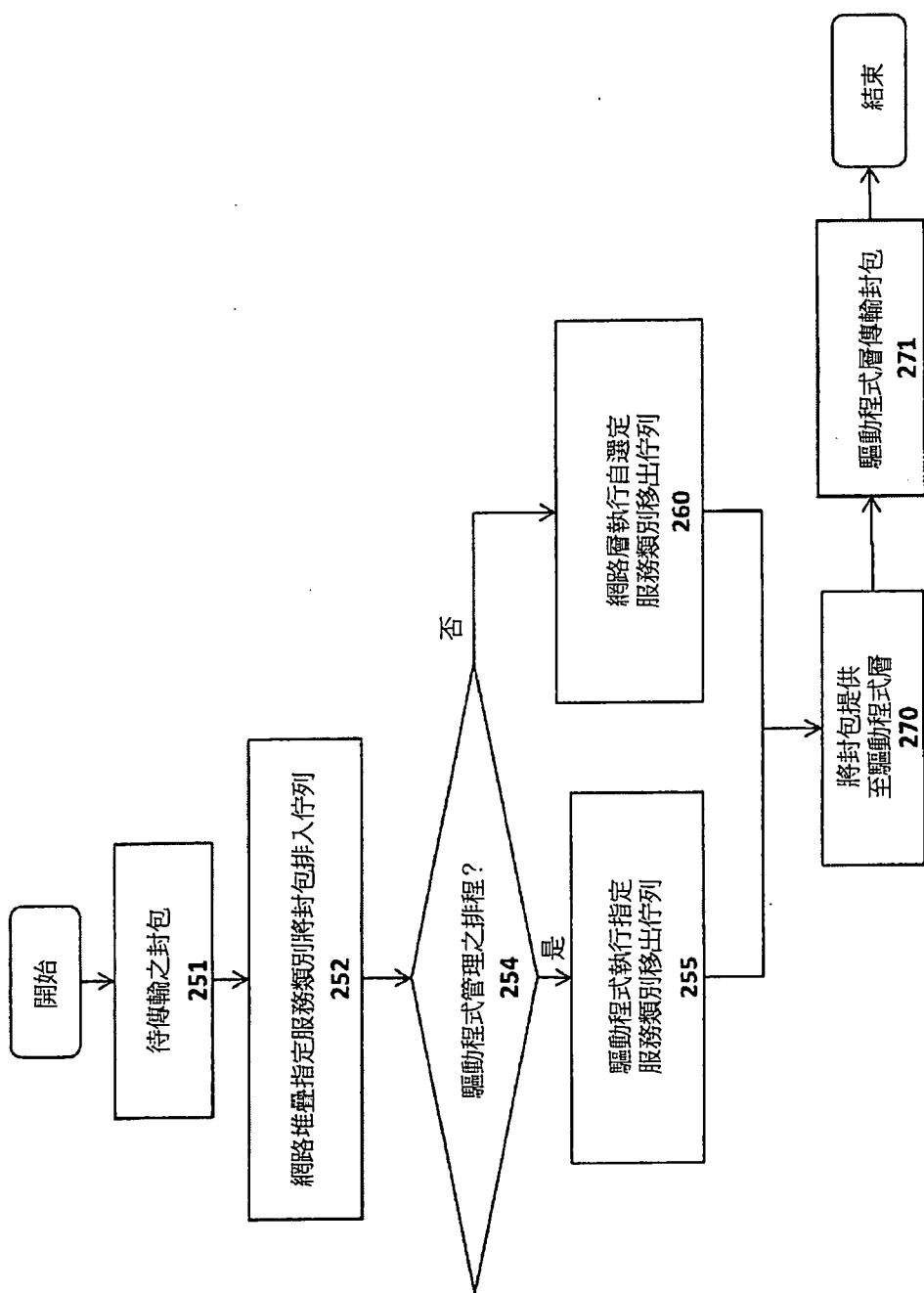


圖2B

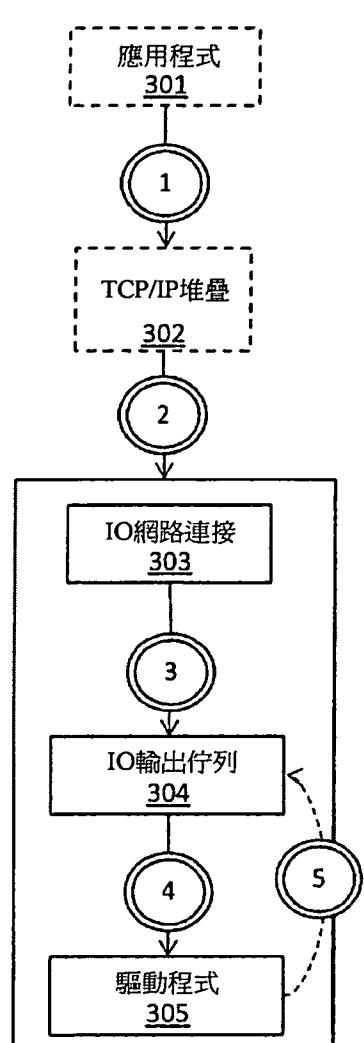


圖3A

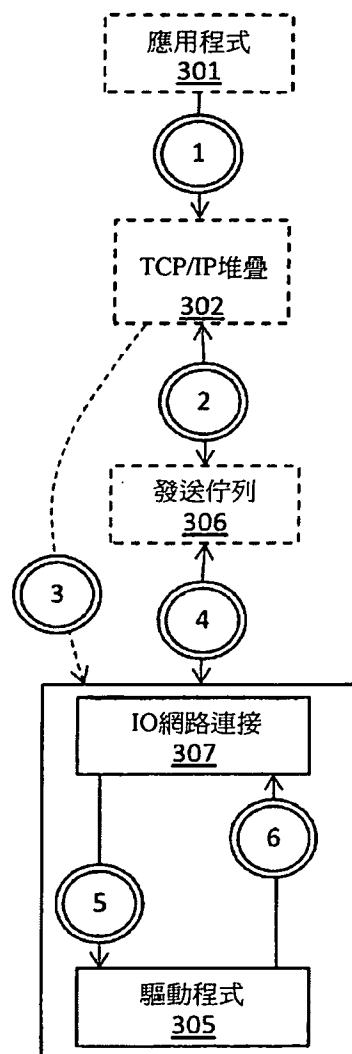


圖3B

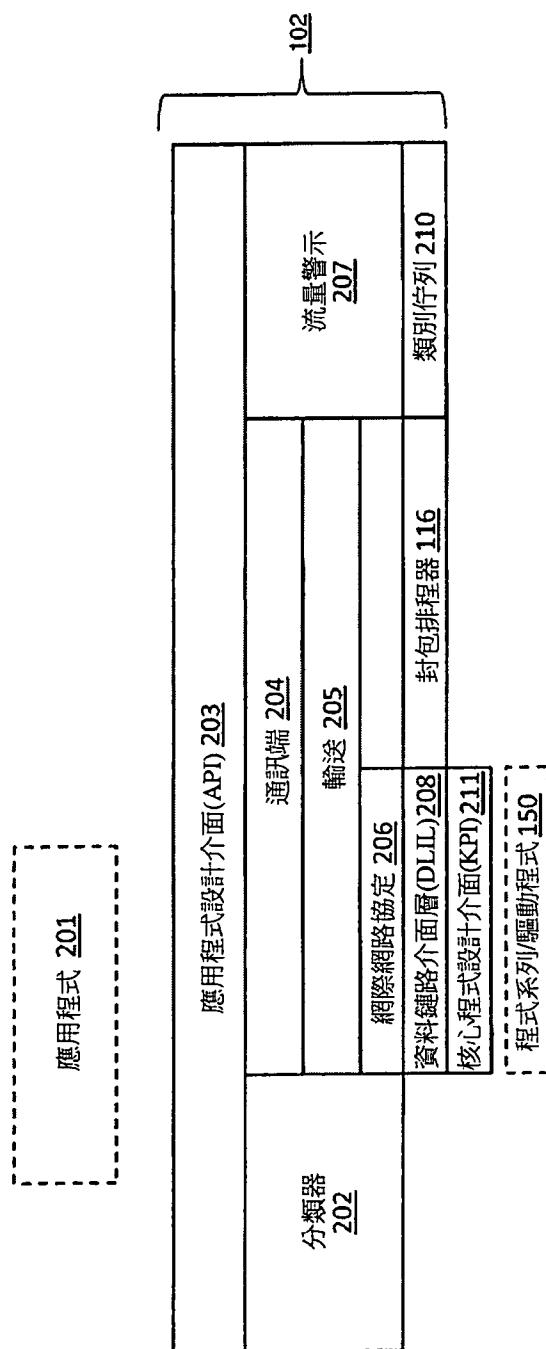


圖3C

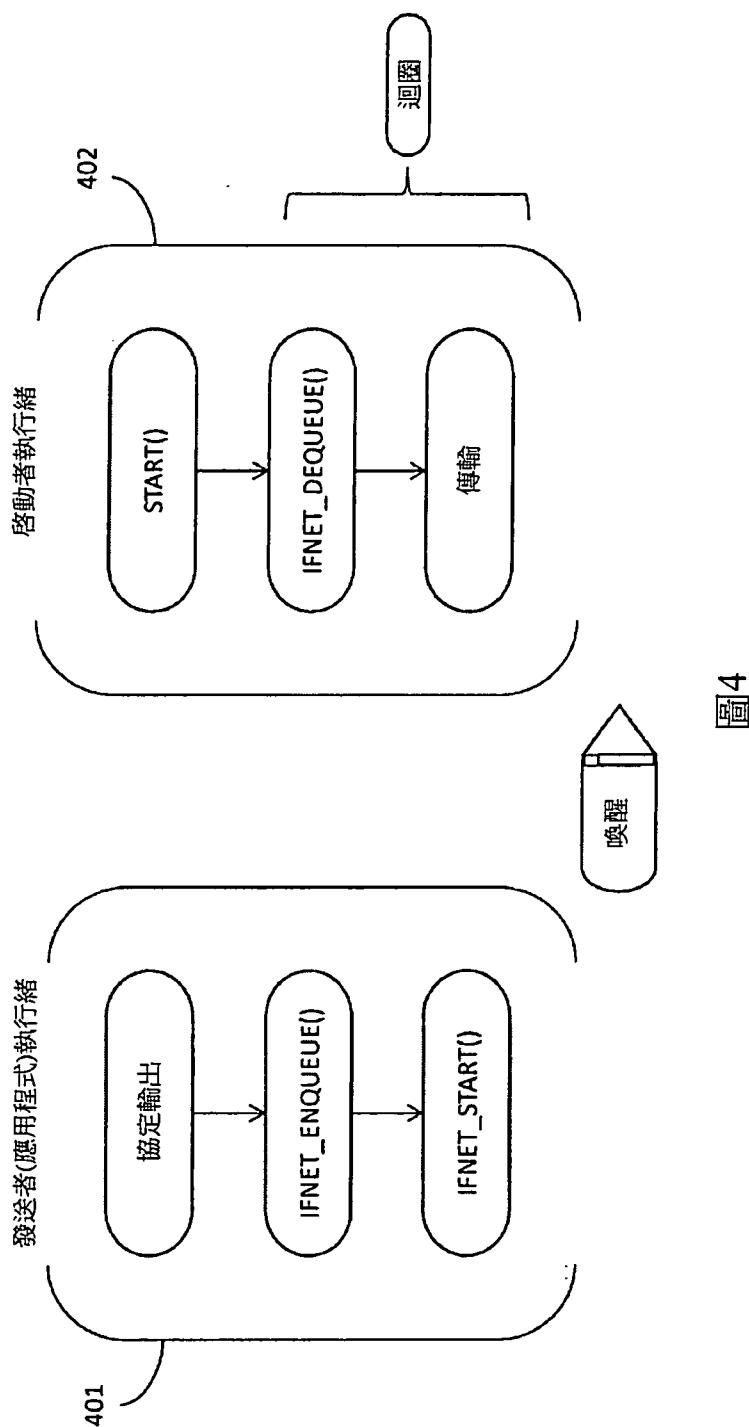


圖4

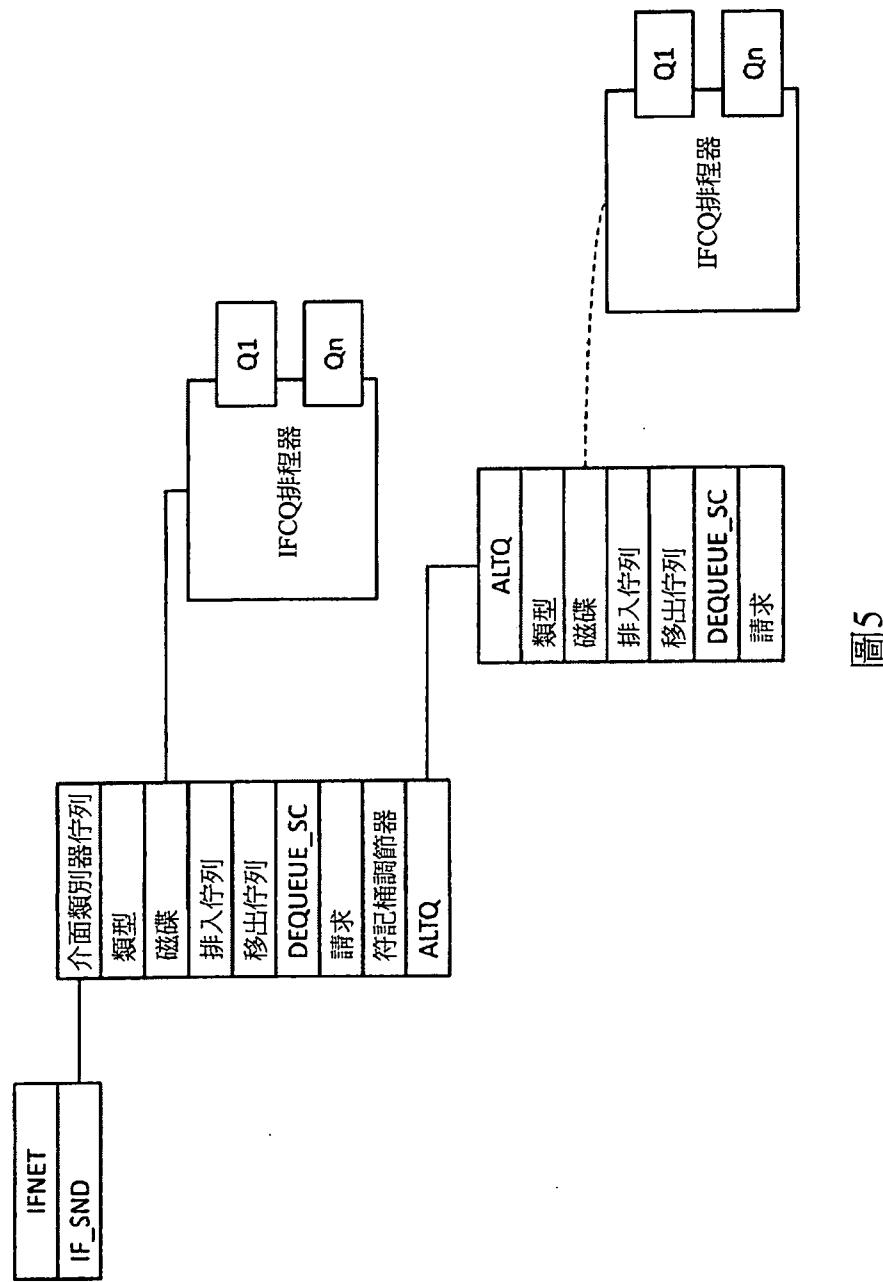


圖 5

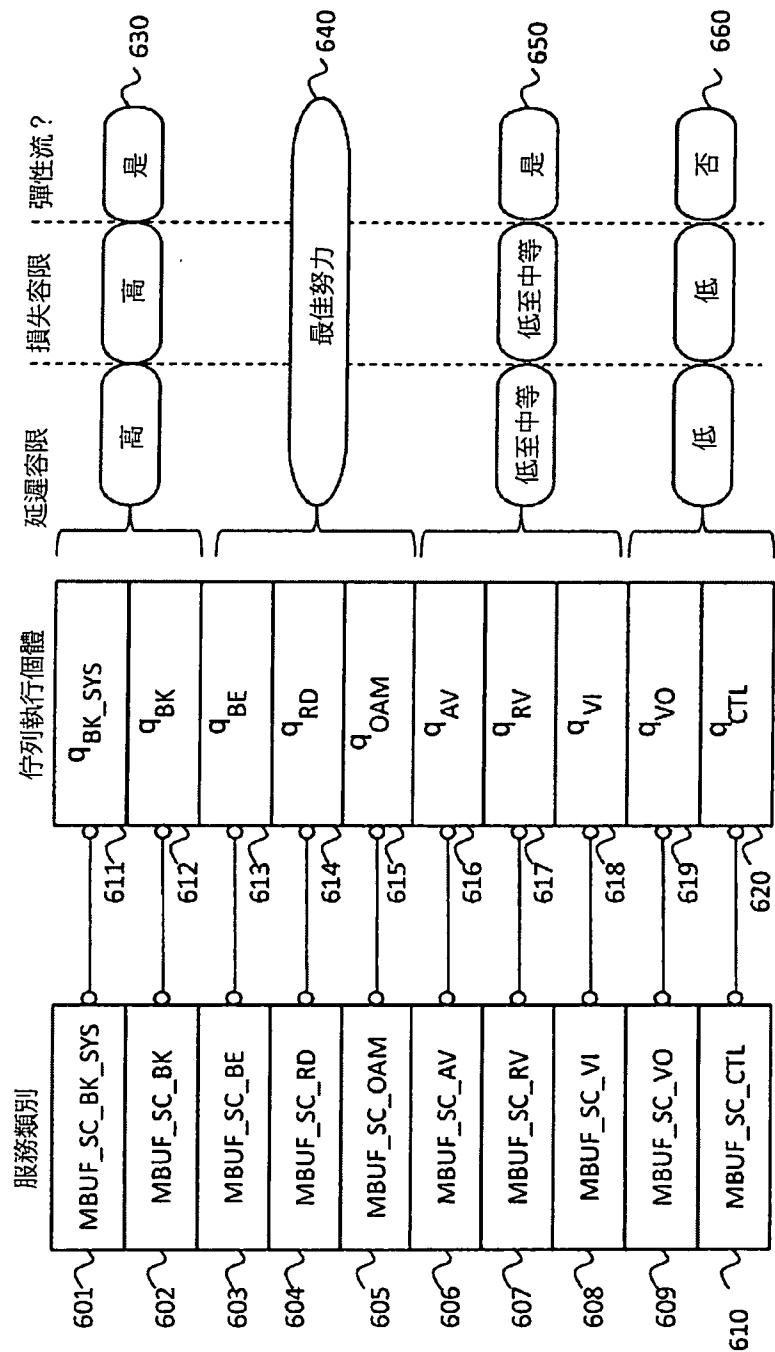


圖6



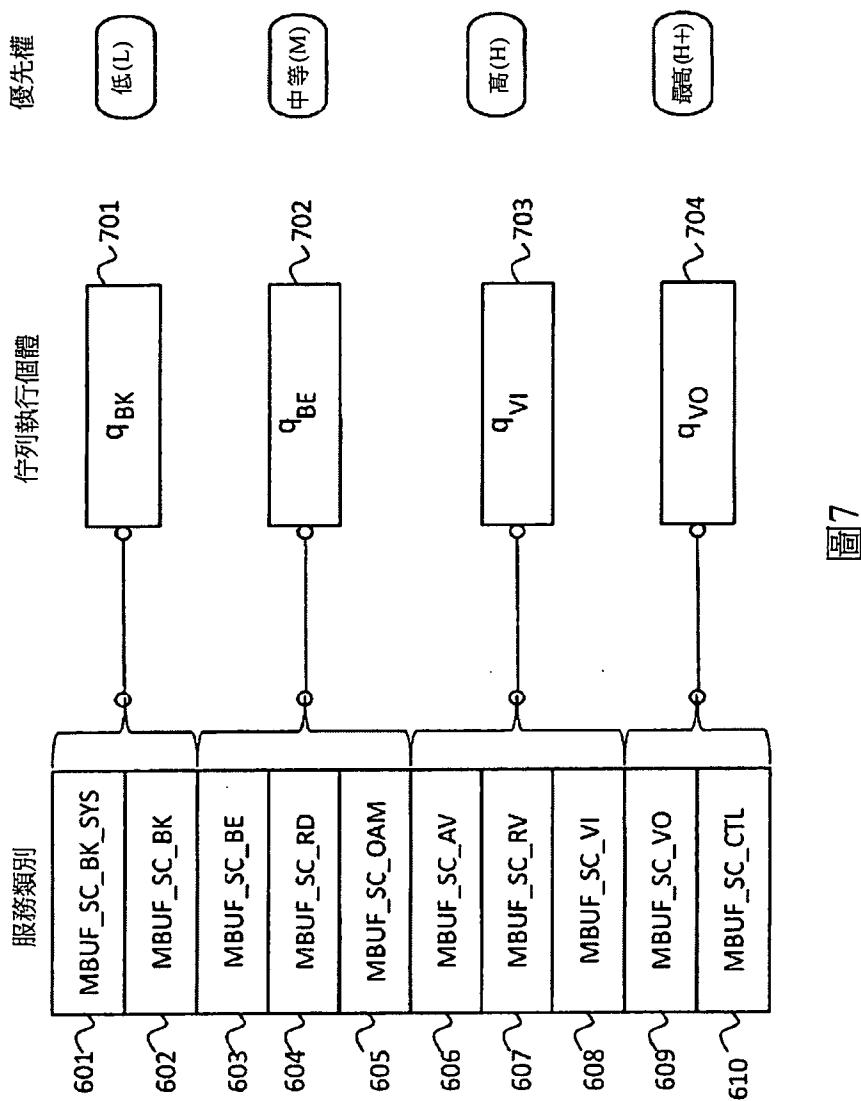


圖7

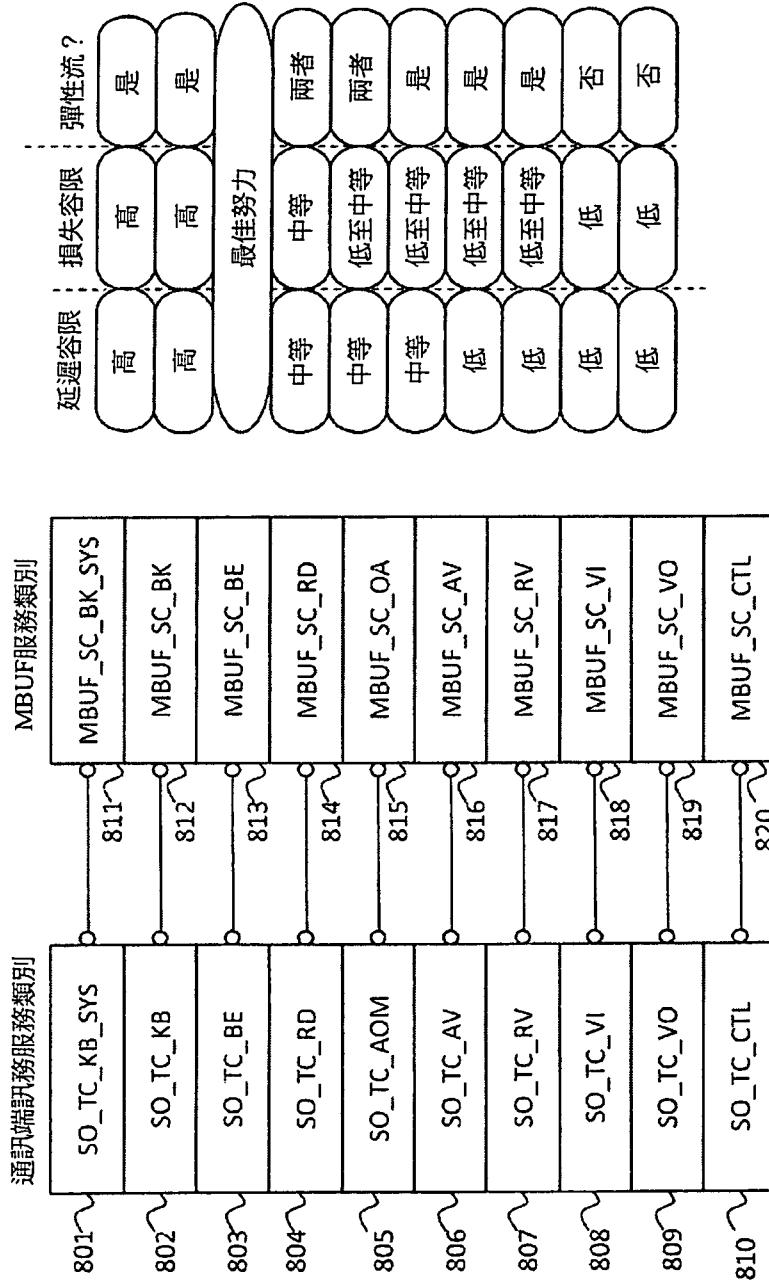


圖8

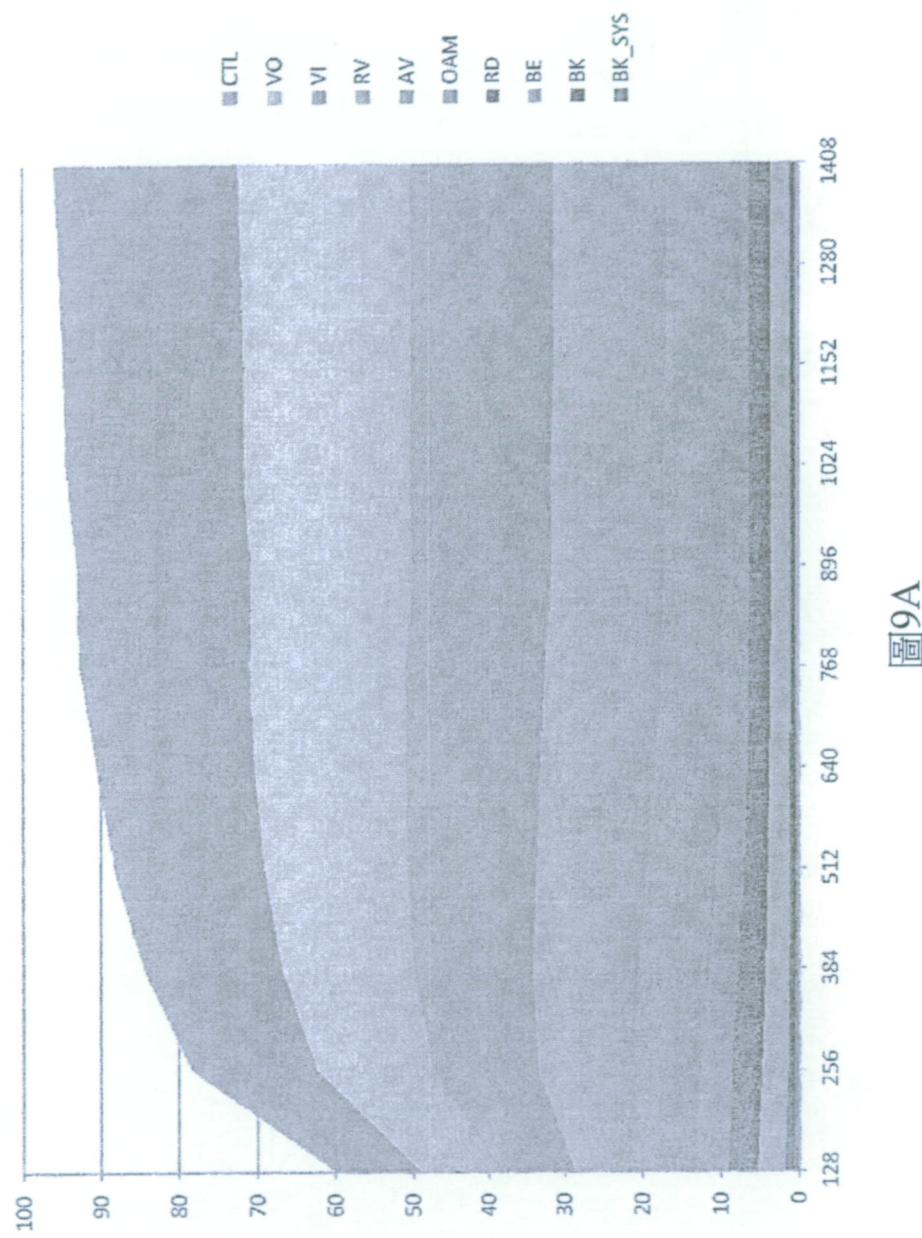


圖9A

```

altq on en1 qfq queue { q_bk_sys, q_bk, q_be, q_rd, q_oam, q_av, q_rv,
q_vi, q_vo, q_ctl }

queue q_bk_sys weight 300 qlimit 256 qfq (lmax 1200 sfb fc)
queue q_bk weight 600 qlimit 256 qfq (lmax 1400 sfb fc)
queue q_be weight 2400 qlimit 256 qfq (lmax 600 sfb fc default)
queue q_rd weight 2700 qlimit 256 qfq (lmax 600 sfb fc)
queue q_oam weight 3000 qlimit 256 qfq (lmax 400 sfb fc)
queue q_av weight 8000 qlimit 256 qfq (lmax 1000 sfb fc)
queue q_rv weight 15000 qlimit 256 qfq (lmax 1200 sfb fc)
queue q_vi weight 20000 qlimit 256 qfq (lmax 1400 sfb fc)
queue q_vo weight 23000 qlimit 256 qfq (lmax 200 sfb fc)
queue q_ctl weight 25000 qlimit 256 qfq (lmax 200 sfb fc)

pass on en1 sc bk_sys queue (q_bk_sys)
pass on en1 sc bk queue (q_bk)
pass on en1 sc be queue (q_be)
pass on en1 sc rd queue (q_rd)
pass on en1 sc oam queue (q_oam)
pass on en1 sc av queue (q_av)
pass on en1 sc rv queue (q_rv)
pass on en1 sc vi queue (q_vi)
pass on en1 sc vo queue (q_vo)

```

圖9B

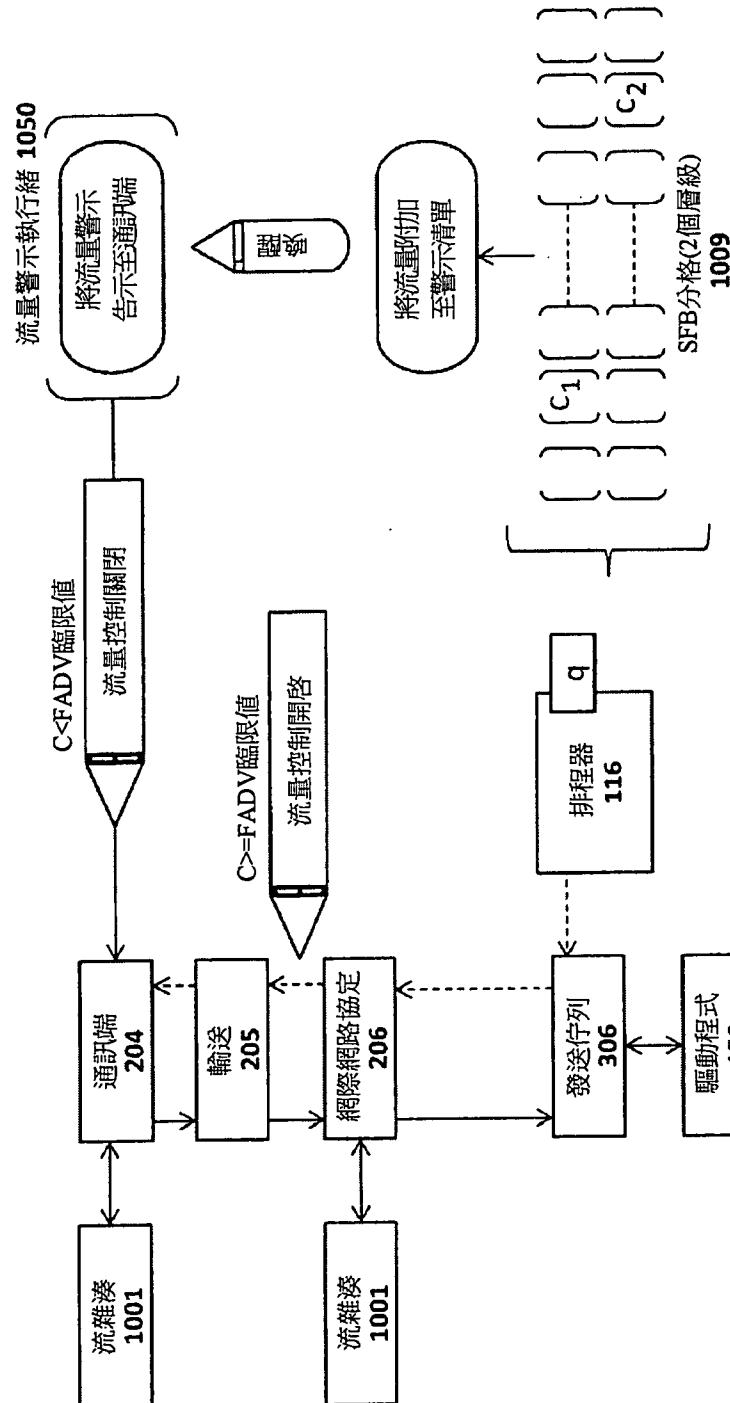


圖10A

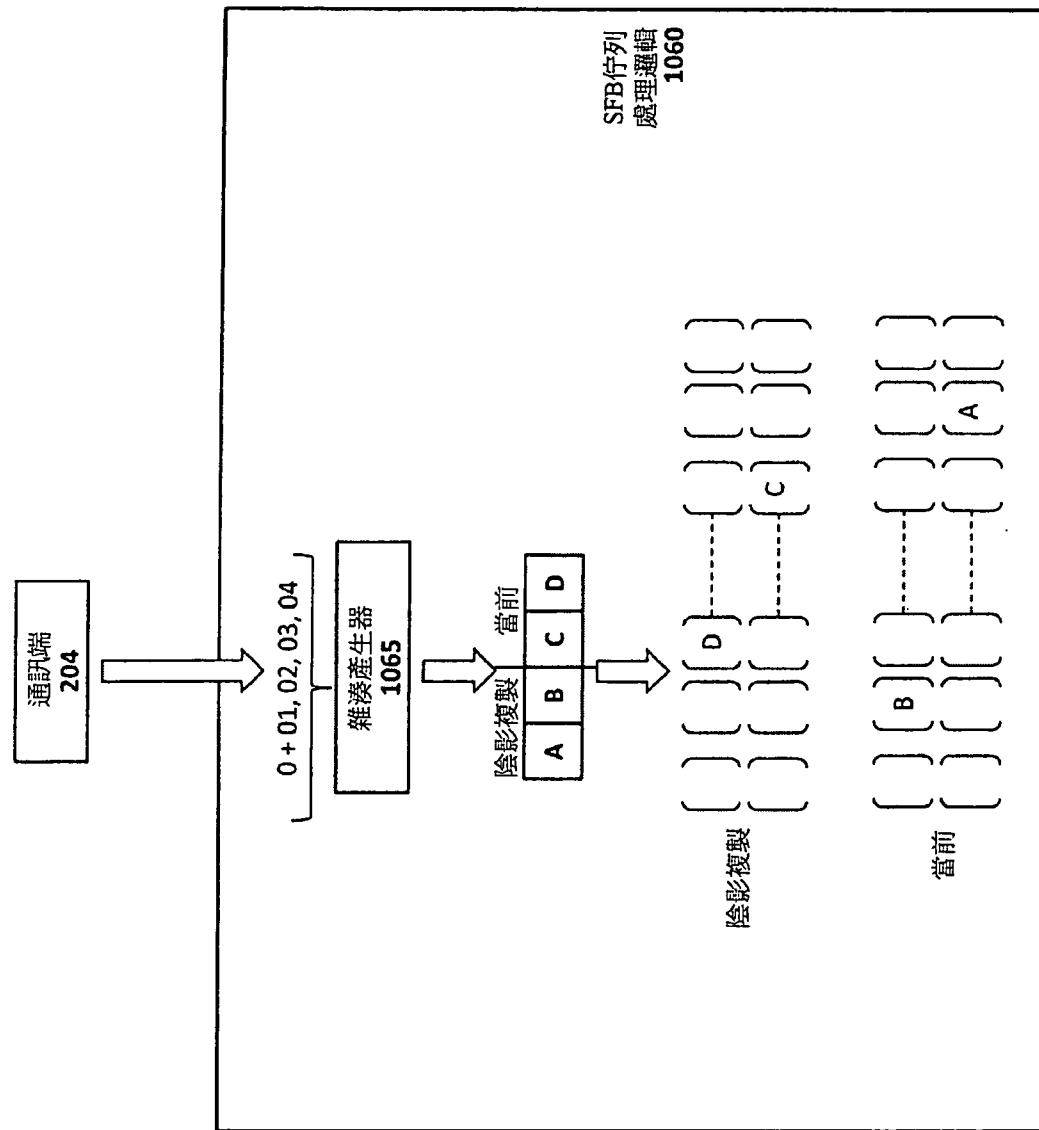
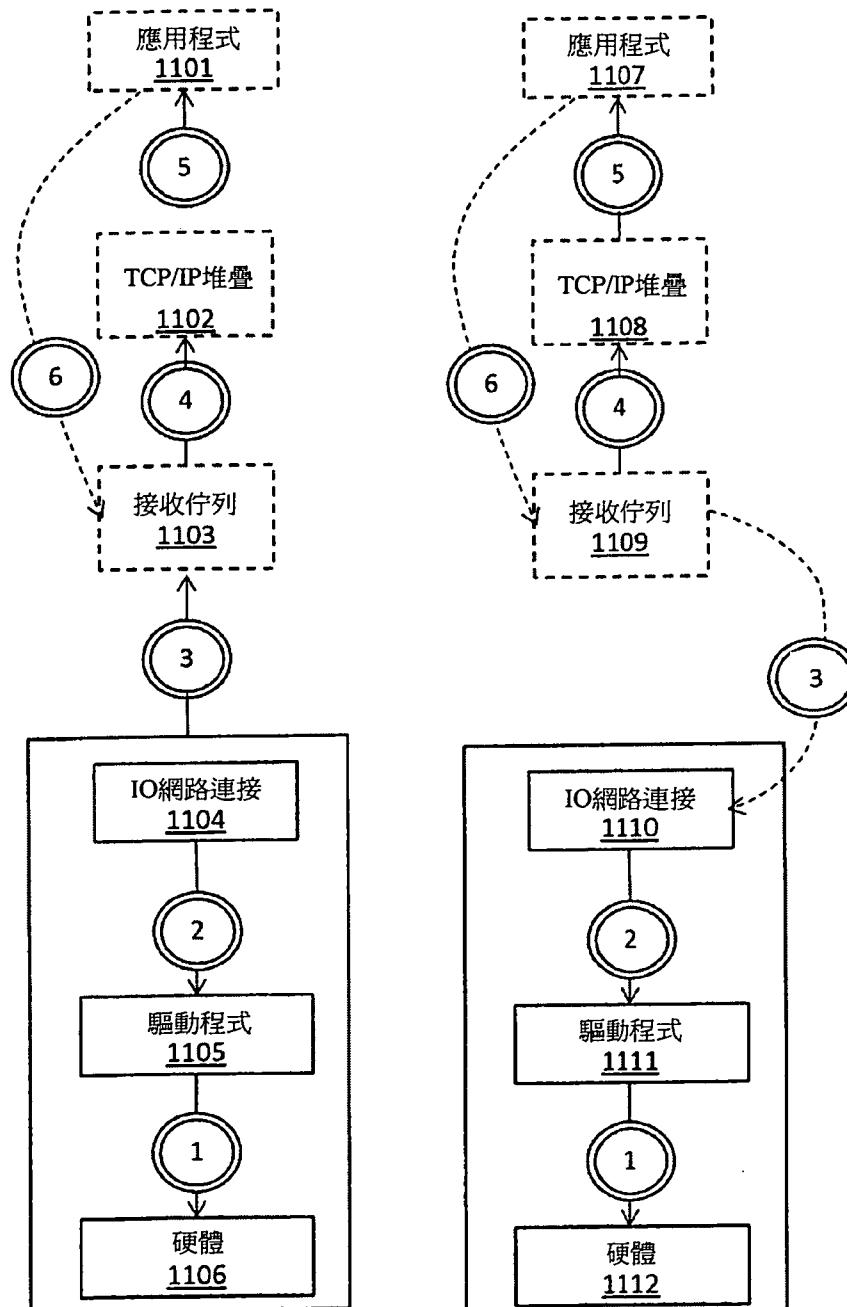


圖 10B



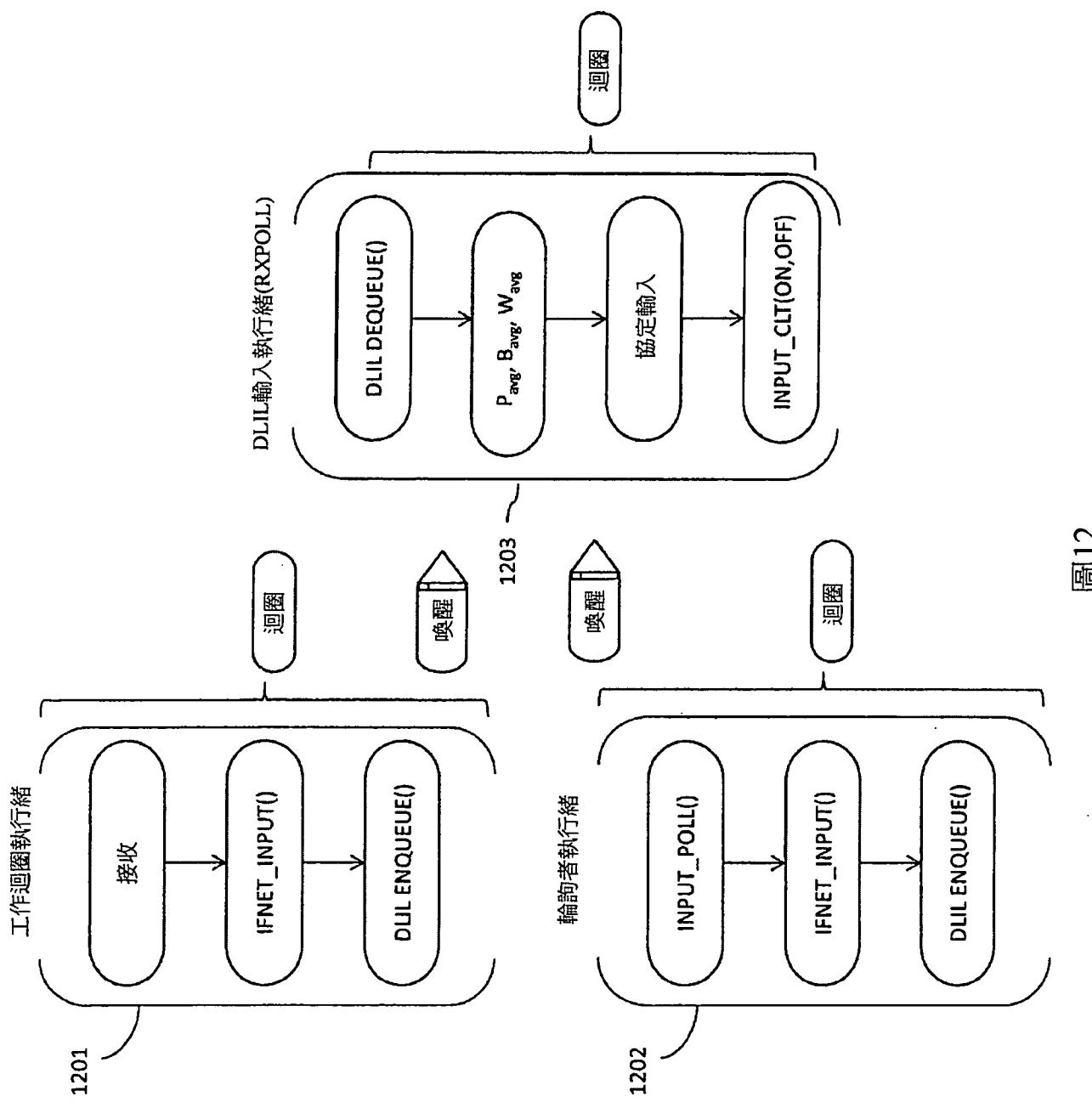


圖12

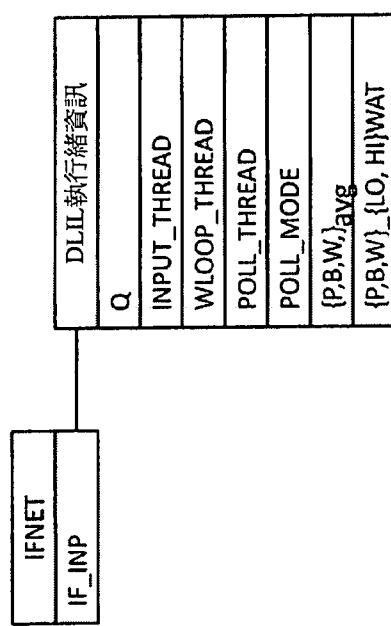
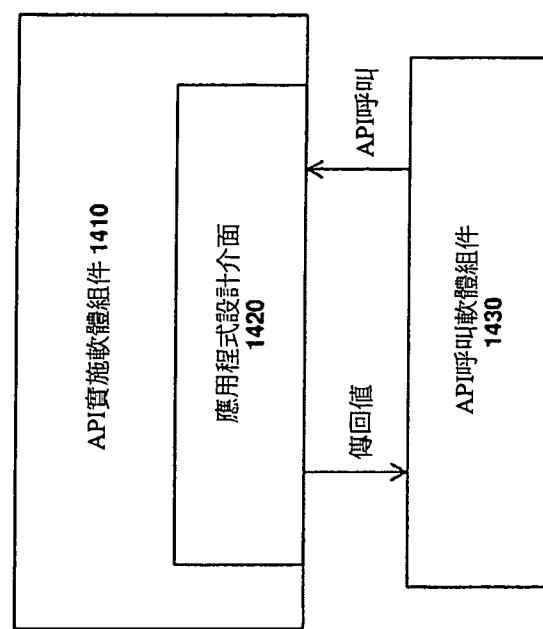


圖13

圖14



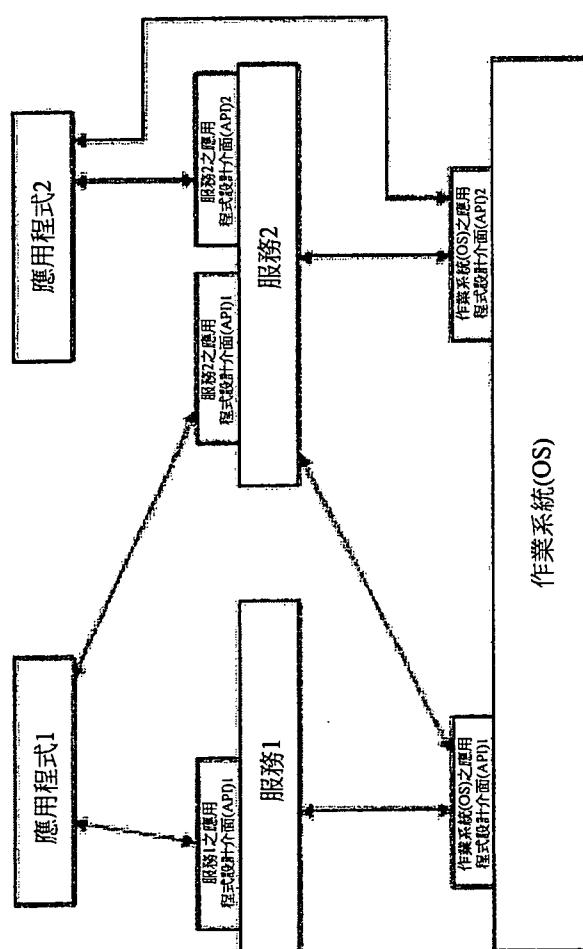


圖15

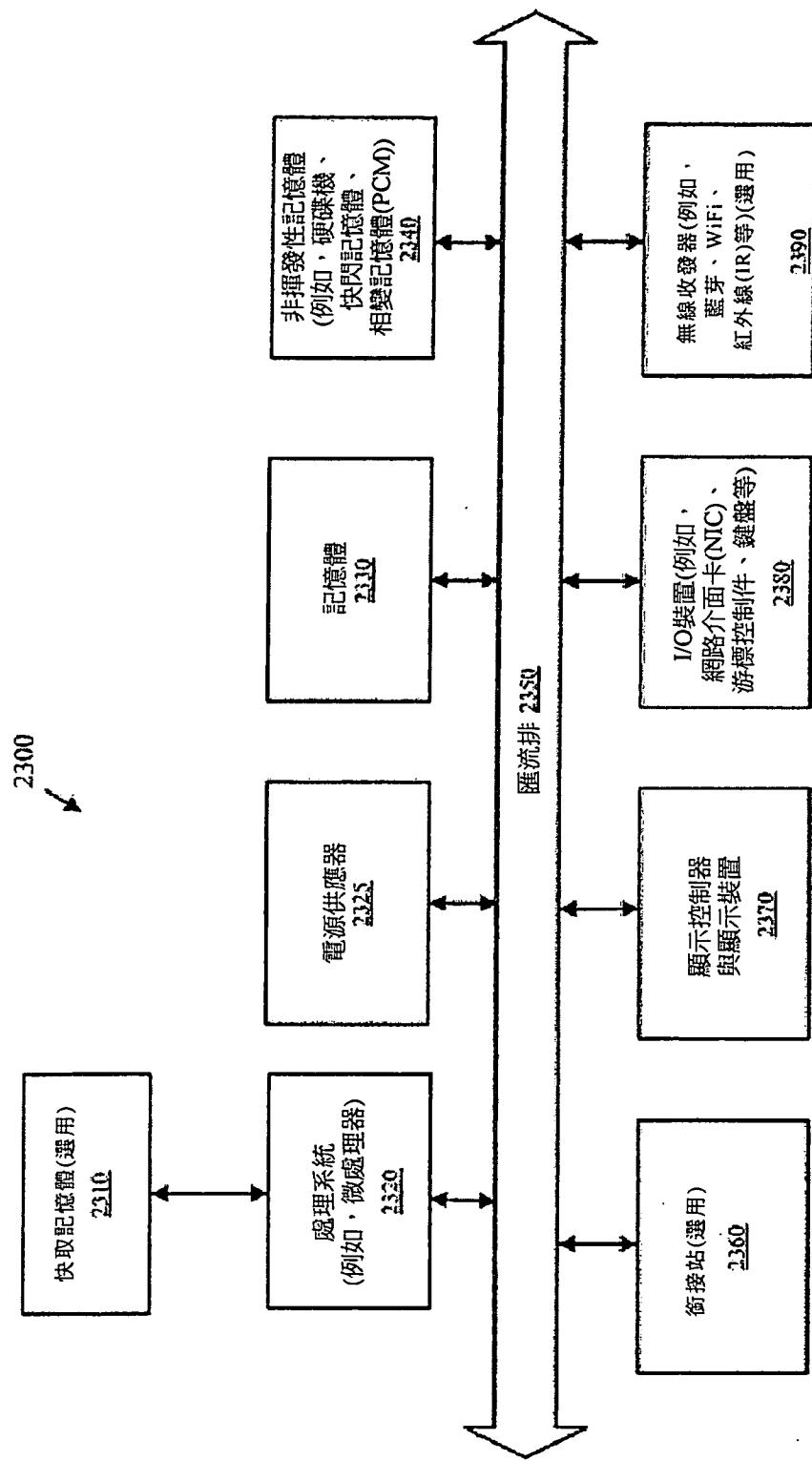


圖16

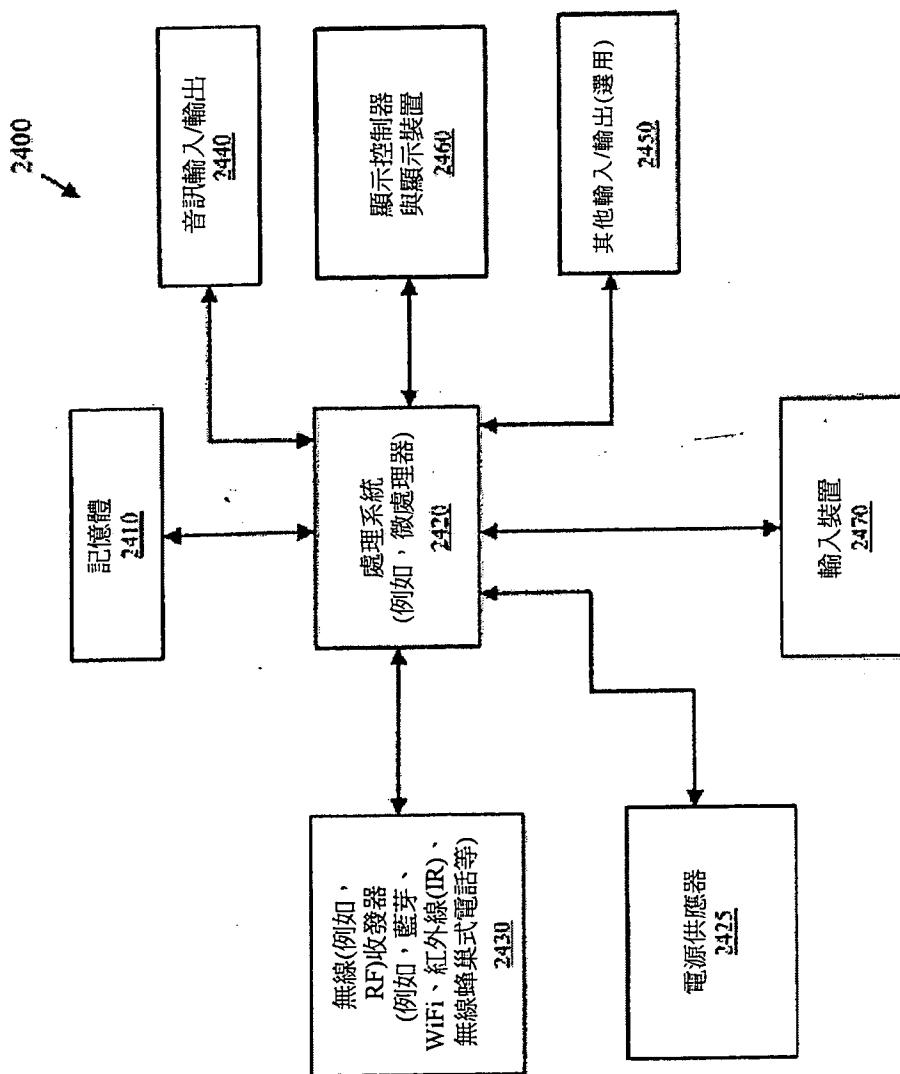


圖17