具 QoS 保證之 Host 端網路介面軟硬體實作

指導老師：侯廷昭

參賽隊員：許宏凱　楊善智　張大偉

國立中正大學電機工程研究所

摘要:

隨著Internet的快速發展，網路服務品質的保證將變成一個重要的課題。我們在這次專題中實作出了具 QoS 功能的主機端網路介面。它具有隨網路壅塞狀況而機動調整其速率的功能，以及提供訊務整流 (Traffic Shaping) 的能力，並且能支援 IP over ATM 協定，與區域寬頻網路接取中心 (GigaPOP) 和國家寬頻實驗網路 (NBEN) 中的 ATM switch 互通。我們使用了創新的設計方法與理論，並經過各種流量狀況的功能與穩定測試，在採用 ATM 為骨幹傳輸技術的網際網路已經逐步成為商業主要的傳輸網路的同時，我們的作品提供了網路服務業者一個良好主機端產品的雛形。

關鍵詞：ABR、IP、ATM、訊務整流、ATM switch、GigaPOP、NBEN。

一、前言

儘管網路的頻寬以驚人的速度成長，我們今天卻仍然發現自己常在壅塞的網路上與別人競爭，而且可預期在不短的未來內也是如此，究其主因係「網路傳輸服務品質不佳」所造成。對於區域網路而言，近來因為 Gigabit Ethernet 技術的研發，區域網路的速度已有相當大幅度的提升，然而這只是把網路壅塞問題轉移到別的地方— 閘道器 (Gateway) 和路由器 (Router) 而已，而且會更加重它們的負擔。針對需要服務品質保證 (QoS) 的廣域網路應用，如一個遠距離的視訊會議 (Video Conference)，遠距教學 (Distance Learning)，以及隨選視訊 (Video on Demand)，目前的 Internet 沒有辦法提供保證的頻寬以及穩定的連線，因為 IP 網路的特性[1]，使得想在上面保證 QoS 非常的困難；而連線式網路如：ATM [2]、ISDN、Fractional T1 卻可以輕易做到，也因為如此，它們現在在骨幹網路中扮演極重要的角色。近幾年來，人們試圖發展新的網路科技，使 Internet 的骨幹網路部分能充分利用連線式網路的優點，達到服務品質保證的要求，如：IP over ATM [3]，多協定標籤交換 (Multi Protocol Label Switching) over ATM [4]，視訊會議協定中的 H.320 (over ISDN)，H.321 (over ATM) [5]，或是直接在IP層中做服務品質保證的 RSVP (Resource Reservation Protocol) [7] 等等。以目前為數眾多的 IP over ATM 網路來看，連線式網路的確是提供服務品質保證的要素。

在主機端 (Host)，當然也要有相對的介面來配合網路。就主機端搭配網路的 QoS 機制而言，它應具有隨網路壅塞狀況而機動調整其速率的功能，以及提供訊務整流 (Traffic Shaping) 的功能。前者適用於對延遲 (delay) 不敏感的數據資料，後者則適用於需保證頻寬的即時資料。

因為以上列出的服務品質保證作法皆可整合入 ATM 網路中，而且 ATM 網路顯然是目前在服務品質保證，系統整合以及廣域網路應用上表現最穩定以及最成熟的網路系統，另外我們也希望將我們的網路介面作品直接整合入國家寬頻實驗網路 (NBEN)，與區域寬頻網路接取中心 (GigaPOP) 中的 ATM 交換機對接，所以我們選擇ATM 網路做硬體平台，而且使用 ABR (Available Bit Rate) Service 結合 ATM 的 AAL5 (ATM Adaptation Layer 5) 來實作 QoS 網路介面。選擇 ABR 服務，是因為 ABR 本身非常適合對數據通訊做流量控制 (Flow control)，若加以巧妙運用，則又可以提供訊務整流 (Traffic shaping) 的功能。目前Internet 網路的訊務型態多是突發性而且沒有固定的流量，也就是說，頻寬的要求通常無法在建立連線時得知，使用ABR 服務則能克服這個問題，它能配合 Internet 網路，給予適當的頻寬，並達到保證最低頻寬的服務品質。

我們所研發的網路介面在硬體方面主要是設計了四顆 FPGA，分別執行 ATM/AAL5 傳送，ATM/AAL5 接收，ABR 控制，和 PCI 控制等功能。在軟體方面，我們開發了 Windows 和 FreeBSD 的驅動程式，以及 ABR 速率計算的演算法。

在創新方面，我們的 FPGA 晶片使用了許多新的設計方式，如：狀態機 (state machine) 設計、

傳送速率的控制，這是設計方法創新；專題中的ABR 服務程式中沒有浮點數的出現，增快了演算法運算的速度，也減低了所需佔用的資源，這也是設計方法創新；專題中的ABR 服務協定使用新的觀念，如：配合硬體創造出新的運算式，這是理論創新；在程式中，由於使用 ABR 服務與我們創造出的介面，不需繁複的設定，即可讓 TCP/IP 架在我們的硬體上，與 Internet 相結合並達到服務品質的保證，如：使用 NDIS (Network Driver Interface Specification) 驅動程式模擬出Ethernet，但無ATM LAN Emulation 的繁複，這是應用創新。

在應用方面，本專題可應用的範圍非常的廣泛，從視訊會議 (Video Conference)、遠距教學 (Distance Learning)、以及隨選視訊 (Video on Demand)、到遠距醫療服務，我們的專題成品皆可以有良好的表現，以及絕佳的品質。在採用 ATM 為骨幹傳輸技術的網際網路已經逐步成為商業主要的傳輸服務網路的同時，我們的作品提供了網路服務業者一個良好主機端產品的雛形。

在完整度上，我們除了研發出完整的硬體與 WINDOWS 95/98/NT 的驅動程式與介面，還研發了 FreeBSD 等其他作業系統的驅動程式，並與 TCP/IP 協定一起做了完整的測試。而且專題成品也經過 Fore ATM switch 與 ATM 協定分析儀各種流量狀況的功能與穩定測試。

這些條件使我們專題成品的創新度，穩定性，效率，以及相容性皆有非常優越的表現。

二、研究目的

我們的專題目的是實作出一套：

- 達到服務品質保證，
- 具備完整且改良的 ABR 流量控制機制，
- 支援 IP over ATM 協定，使其能與 Internet 結合，
- 具有訊務整流 (Traffic Shaping) 能力，
- 使其能與區域寬頻網路接取中心 (GigaPOP) 和國家寬頻實驗網路 (NBEN) 中的 ATM switch 互通，
- 具有能支援多種 IP 層服務品質保證協定的擴充性，
- 支援多條連線 (connection) 的軟硬體介面。

我們發現要完成這樣的網路介面，有著相當的難度，這類題目在國內學術界實作的人並不多。而中正電機所已致力於發展 ATM 網路介面多年，也曾於八十六年度通訊競賽獲獎，只是之前的作品只做到 ATM 層與 ATM 適應層 (AAL5)，我們則更進一步利用先前 ATM 和 AAL5 的基礎，全力投入於 ABR 和 QoS 的實現。

若將我們的實作與之前研發出來的 ATM 網路介面相比，我們不僅加入新的機制，ABR Service，也使用了新的硬體設計方法與新的軟體設計流程，改良了先前的 ATM 和 AAL5 模組，使其無論效率與功能應用上皆有新的突破與創新。

目前僅國外極少數的公司有類似功能的產品，在 ABR 服務功能上，他們幾乎是大同小異的使用了舊的機制與製作方法，進一步來說，目前的 ATM 網路卡皆是只符合 ATM Forum Traffic Management Specification 4.0 的建議，而相對於他們的產品，我們卻更進一步符合 ATM Forum Traffic Management Specification 4.1的規範，另外，我們強調 ABR 服務的硬體與軟體的理論與設計方法創新，我們除了有更新的設計，並能使其在有限的硬體上發揮最大的效益。

針對 ABR 的實現而言，究竟它的處理應擺在網路介面(硬體)或是主機端(軟體)，我們原先的構想是擺在網路介面。這主要因為 ABR 流量控制是 ATM 層的機制，既然我們把 ATM 的動作全部在網路介面用硬體處理掉，很自然的，ABR 的功能也最好能以硬體完成之。但是經過我們事前仔細的評估，發現 ABR 計算居然需要一顆專屬的高性能微處理器才能達成，就經濟效益而言，甚不可行。因此我們將計算繁複的部份交由主機驅動程式 (driver) 來做，RM cell 的處理則交由 ABR Controller chip來做。基於如此之構想，我們提出一創新的簡化 ABR 速率計算的方法 (細節於第四節陳述)。

三、原理與分析

我們的網路介面使用的原理如下：

(一)、網路協定堆疊：

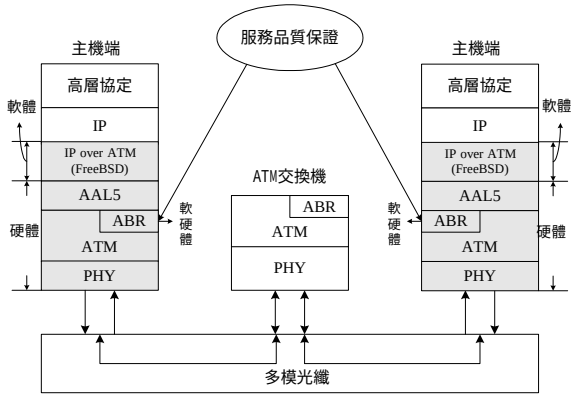


Fig.1 網路協定堆疊

這套網路介面可運作在一般有 PCI 介面的主

機上，作業系統可以是 Microsoft Windows 95 / 98 / NT, 或是 FreeBSD 2.x ~ 4.x, 作業系統上的 Internet 應用軟體完全不會受到底層硬體不同的影響。它所提供的品質服務保證由 ABR 服務達成。

(二) 、Available Bit Rate (ABR)

ATM 技術最大的好處，就是它能支援非常多樣化的的交通流量形式，並能符合區域網路或廣域網路環境中各式服務品質保證的需要。這項功能是由一些訊務管理機制來達成，這些機制使得 ATM 網路的運作更加的有效率與穩定。最主要的訊務管理機制有連線允入控制 (CAC) ，使用參數控制 (UPC) ，優先權控制, 訊務整流等等。在這些之中，壅塞控制和訊務整流最有效但也對網路和系統的設計人員有著最大的挑戰性。

在 ATM 網路上, 有提供兩種不同的壅塞控制策略，根據不同的服務型態而使用開放性迴路或是封閉性迴路。第一種適用在 CBR 或 VBR，他們可使用的頻寬在網路連線建立時即已設定完成，我們稱這種迴路為預先式壅塞控制。但是在 Internet 網路中，頻寬的要求通常無法在建立連線時得知，使得開放性回路並不適用於這類服務，進一步來說，尖峰速率 PCR (Peak Cell Rate) 設得低，則可能在某些狀況無法滿足應用程式需求，而設得高又會在大部分的時間裡無法利用到滿載的頻寬，這將造成極低的網路使用效率。為了能兼顧 ATM 網路的優點並與 Internet 相結合，我們使用了另一種的服務型態 – ABR 服務。

ABR 服務使用一套 rate-based 的流量控制機制，去控制資料來源端的速度，實際控制的方法則是改變 ABR 演算法中的變數，迴授模式使用 RM (Resource Management) Cells 傳遞至資料來源端來達成。這樣做的目的是期望終端系統能根據迴授的資訊動態的調整速度。ABR 服務的傳輸模式有兩種：

A 、單向傳輸模式

在此模式中，網路卡 1 是傳送端，網路卡 2 是接收端，只有網絡卡 1 要送資料給網路卡 2。

網路卡 1 在傳送 data cells 的期間會穿插傳送 Forward RM cell，網路卡 2 收到 Forward RM cell 後，會將網路狀況寫入 RM cell, 並傳送 Backward RM cell 給網路卡 1。如果網路卡 2 收到 data cells, 它會將 data cells 重組回原來的 packets。如圖 2 所示。

當網路卡 1 收到返回的 Backward RM cell, 它會依據此 cell 得知網路的狀況, 並調整至適當的傳送速率。

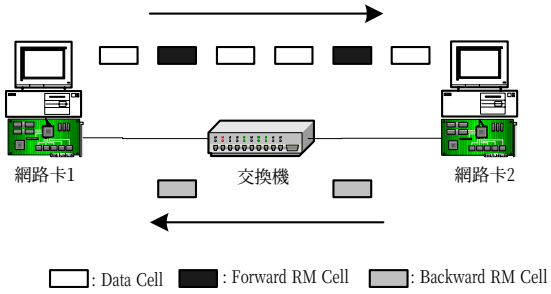


Fig.2 單向傳輸模式

B.雙向傳輸模式

在此模式中，網路卡 1 和網路卡 2 都是傳送端，也都是接收端。傳輸模式跟單向的大致相同，只是網路卡 2 亦有資料要送給網路卡 1，這會影響到 data cell， Forward RM cell 和 Backward RM cell 傳送的優先權，而優先權是由 ABR Controller 晶片的邏輯設計所決定。

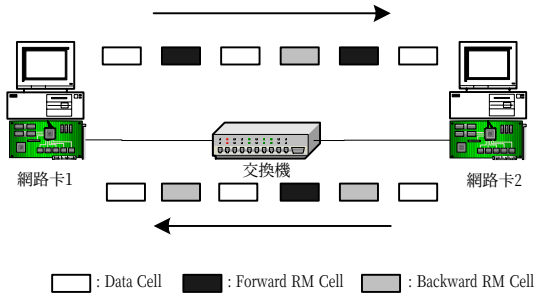


Fig.3 雙向傳輸模式

(三) 、ATM 網路主機端軟硬體介面

我們設計了一張標準 PCI 介面的 ATM 網路卡，使之可以插在主機 (host) 內。此 ATM 網路卡支援 ABR 服務，可同時為傳送端和接收端。在我們的設計中，實作了兩顆晶片 — Tx 及 Rx, 負責資料的傳送及接收。另外，為支援 ABR 服務的機制，特別設計了 ABR Controller 的晶片，來控制傳送的速率。

為了要讓我們設計的晶片能穩定且有效的支援 ABR 服務，與處理多樣的 cell，我們在硬體上使用了相當多的創新設計方法，如：狀態機 (state machine) 設計、傳送速率的控制、實現資料管理 (data management) 等等, 而我們在第四節有詳細的描述。

當網路卡做為傳送端時， Tx 晶片會在 data cell 傳送期間，依照 ABR 演算法中設定的時間，固定發送 Forward RM cell, 當接收端收到 Forward RM cell，會將此 cell 變成反向 (Backward) RM

cell 返送回去給傳送端。

當傳送端的 Rx 收到 Backward RM cell ，主機 (host) 會依據此 cell 所載送的資訊來更新速率，因為 RM cell 在來回時收集了交換機和接收端的擁塞資訊。

當網路卡做為接收端時，只要 Rx 晶片收到 Forward RM cell，系統會依照 ABR 演算法寫入壅塞資訊到 RM cell 中，然後再由 Tx 晶片返送 Backward RM cell。

進一步來說，在主機端收到 RM Cell 後，我們用一套演算法調整傳送端的可利用細胞速率 (ACR) 值，ACR 值直接影響了網路介面送出資料的速率。

另外 RM Cell 除了是由主機端發出外，也可能是交換機 (Switch) 因網路狀況主動發出的，端看交換機有無支援 ABR 服務中的功能。

ATM Forum Traffic Management 標準中對調整 ACR 的演算法定義了相當多的浮點變數 (Floating point variable)，並且有大量的浮點乘除運算。這些運算不適合擺在硬體，因為會為此付出相當昂貴的硬體成本，而且整個演算法也變得無法彈性更改。但是它也不適合擺在軟體，在 Microsoft Windows 系列以及大部分的作業系統中，都強烈的建議或是禁止核心的驅動程式 (Kernel driver) 使用浮點運算，因為浮點運算完全是跟著主機的 CPU種類不同而有不同運作方式，此種硬體依賴 (Hardware dependent) 的程式會使作業系統安裝到其他的硬體上 (porting) 時造成不可預期的錯誤。

目前國外有支援 ABR 服務的 ATM 介面，有的是耗費昂貴的代價，將 ABR 服務做在硬體；有的是將浮點運算擺在軟體，而使用查表的方式，這種方法雖然避免了上述問題，卻使得驅動程式龐大而且耗用主機資源。

所以我們改寫了整個 ABR 演算法，使它有相同的功能卻是不同的作法，我們將一部份功能放在硬體，一部份放在軟體，最重要的是整個 ABR 演算法沒有浮點變數與浮點運算，這在接下來的第五節也會有詳細的描述。

四、軟硬體系統

(一)、 ATM 網路卡的硬體架構

在我們設計的系統中， Tx 必須對 ABR Controller 的請求做出適當的回應。它實現了 ATM AAL5 和 ATM 層關於傳送方面的協定， Tx 主要的功能就是把上層的封包切成 ATM data cell，並能組成各種的 RM cell，包括 Forward RM cell， Backward RM cell 以及 out-of-rate Forward RM cell。

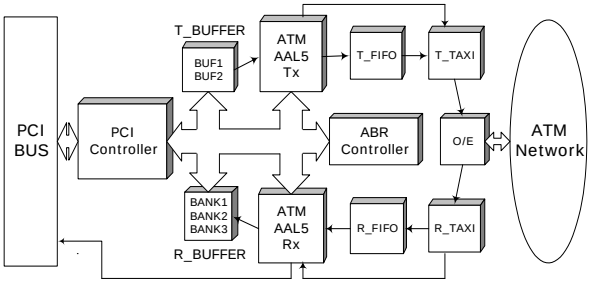


Fig.4 系統架構

T_BUFFER (IDT7203) 被分為兩個部份，BUF1 和 BUF2。它們各別是由兩顆 8-bit 的 FIFO 所組成，針對自己的連線，儲存上層所送下的封包。T_BUFFER 也有緩衝主機和 Tx 速率差異的功能。

T_FIFO (IDT7201) 是用來緩衝 Tx 和 T_TAXI 速率的差異。

T_TAXI (AM7968) 是屬於傳輸的實體層晶片，它是 Tx 和傳輸媒介的介面。

Rx 晶片實現了 ATM AAL5 和 ATM 層關於接收方面的協定，這顆晶片主要的功能就是接收 ATM cells。當收到的是 RM cell，它會檢查它是哪一種 cell，假如是 Forward RM cell，網路的擁塞資訊會被更新，並將 Forward RM cell 的負載傳送到 Tx 晶片，以做為 Backward RM cell 傳送的資料。ABR Controller 也同時被告知收到一個 Forward RM cell 了。如果是收到 Backward RM cell，它會透過中斷的方式來通知主機。而當收到的是 data cell 時，它會在 R_BUFFER 中重組成原來的封包，只要封包完整的組完，它亦會中斷主機。

R_BUFFER (IDT7203) 是 Rx 用來存放並重組封包的地方。它包括了三個部份，BANK1，BANK2，BANK3。其功能跟 T_BUFFER 差不多。

R_FIFO (IDT7201) 是用來緩衝 Rx 和 R_TAXI 速率的差異。

R_TAXI (AM7969) 是屬於傳輸的實體層晶片，它是 Rx 和傳輸媒介的介面。

Backward RM cell 攜帶從交換機和接收端收集到擁塞資訊，傳送端依此來更新 ABR 的參數，並調整傳輸速率。由於在 ATM Forum Traffic Management 標準中，有些 ABR 參數須要浮點運算，因此這部份交給具高運算效率的 CPU 來處理，

以簡少硬體的成本，並增加 ABR 參數運算的靈活性 (此部份常被更改，從 ATM Forum Traffic Management SPEC. v4.0~4.1 中就可發現)。

ABR Controller 晶片能接受從主機送來的 ABR 參數，並依此參數來調整要求 Tx 傳送的時間間隔，達到了控制傳送速率的功能。此外， ABR Controller 還包含了有關 ABR 流量控制的邏輯設計，來決定此刻要傳送的是 data cell， Forward RM cell， Backward RM cell 或是 out-of-rate Forward RM cell，此部份是參考 ATM Forum Traffic Management SPEC. v4.1，並考慮實際硬體情況以及 ABR 整體效率來改良的。

A. Tx 晶片運作流程

當網路卡啟動 ABR 模式，Tx 會等待 ABR Controller 的要求 (request) ，要求分為以下幾類：

・Data_Send

CRC32 值和傳輸長度 (TL) 兩者會被載入，TL 是個重要的值。當 TL 大於 40 時，COM (Continuation of Message) cell會被傳送；否則，EOM (End of Message) cell會被傳送。

— COM cell 傳送過程：

如果 TL 大於 48 ，COM cell 就不包含 AAL5 的填補 (PAD) 欄位。在 cell 送出之後，CRC32 值會被儲存，而 TL 會被減去 48，並通知 T_TAXI 傳送。如果 TL小於 48 而大於 40 ，當每個位元組送出之後，TL 會減去 1，直到 TL 為 0。此時填補位元組接在 data 後面，以形成一個 cell。當這個 cell被送出去之後，CRC32 值會被儲存，並通知 T_TAXI 傳送。

— EOM cell 傳送過程：

假如 TL 不等於 0，當每個位元組送出去之後， TL 會減去 1，直到 TL 為 0。然後加上填補位元組後，再加上 AAL5 標尾 (trailer)。於是整個封包就完整的送出去了，之後便通知 T_TAXI傳送。

・RM_Send

RM cell 的要求可分為以下幾種情況：

— Forward RM cell 傳送過程：

Tx 晶片暫存 7 個位元組的最新 ABR 參數，分別是 1 位元組的訊息欄位，2 位元組的明確速率 (Explicit Rate) 欄位，2 位元組的現在速率 (Current Cell Rate) 欄位，2 位元組的最小速率 (Minimum Cell Rate) 欄位。當收到此要求，Tx 晶片會將這7個位元組包裝成一 Forward RM cell 送出。

— Backward RM cell 傳送過程：

Tx 晶片另外暫存了 7 個位元組的 Backward RM cell ABR 參數，它們是由 Rx 晶片收到 Forward RM cell 時寫入到 Tx 晶片上的，當 Tx 晶片由 ABR Controller 收到此要求， Tx 晶片會將這 7 個位元組包裝成一 Backward RM cell 送出。

— out-of-rate Forward RM cell傳送過程：

當Tx收到此一要求，它除了會將這7個位元組包裝成一 Forward RM cell 之外，還會把 ATM cell 標頭 (header) 中的 CLP (Cell Loss Priority) 設為 1，並送出。

上述的運作流程如下圖所示。

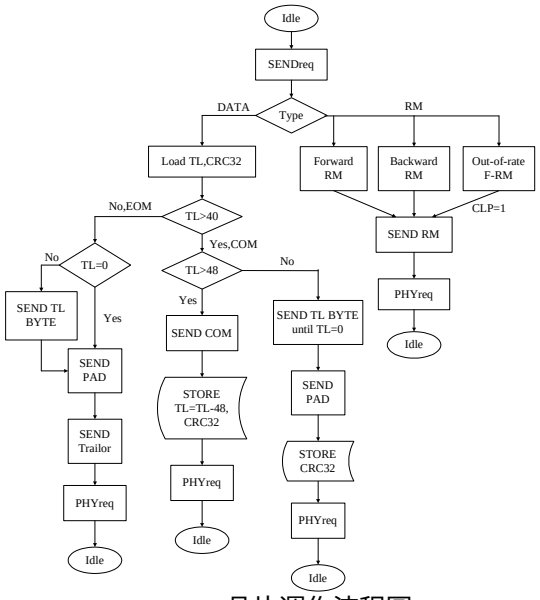


Fig.5 Tx晶片運作流程圖

B. Rx 晶片運作流程

這顆晶片主要的功能就是從網路上接收 ATM cells，並針對不同型態的 cell，做不同的處理。當一個新的 cell 收到後，首先它的標頭會被檢查，以判斷它屬於哪個連線 (connection) 和它的正確性，它的負載種類也會被決定。

・如果是 data cell

如果它屬於某個 CAM 中已註冊封包的一部份，其中 CAM 是來記錄目前三個 R_BUFFER — BANK1，BANK2 和 BANK3— 的狀態，它會找到那個 CAM entry；否則，它會找到一個沒有被佔用的 CAM entry。如果沒找到，它就會被丟棄。

CAM	Found	Free	VC	Length	CRC32	Cok	CINT
1							
2							
3							

Fig.6 CAM Table

— 如果是 COM cell：

目前暫存的 CRC32 值會載入和新的負載一起計算， 48 位元組負載會寫入至 R_BUFFER。在負載完全寫入後，新的 CRC32 值會被記錄下來。

— 如果是 EOM cell：

目前暫存的 CRC32 值會載入和新的負載一起計算，負載會寫入至 R_BUFFER，封包長度會由負載中取出。在負載寫入至 R_BUFFER 後， CRC32 會被檢查，假如檢查是正確的話，主機會被通知，否則會將封包丟棄，而且被佔用的 CAM entry 會被釋放掉。

・如果是 RM cell

— 如果是 Forward RM cell：

ABR Controller 會被通知。除了 CRC10 欄位外，整個負載都會做 CRC10 的計算。DIR，CI 欄位會被更新，Message，ER，CCR，MCR 欄位會存起來，並傳到 Tx 晶片。

如果整個 cell 處理完成，CRC10 就檢查好了。假如 CRC10 檢查是正確的， ABR Controller 會再被通知一次，表示允許 ABR Controller 送出 Backward RM cell 的請求。

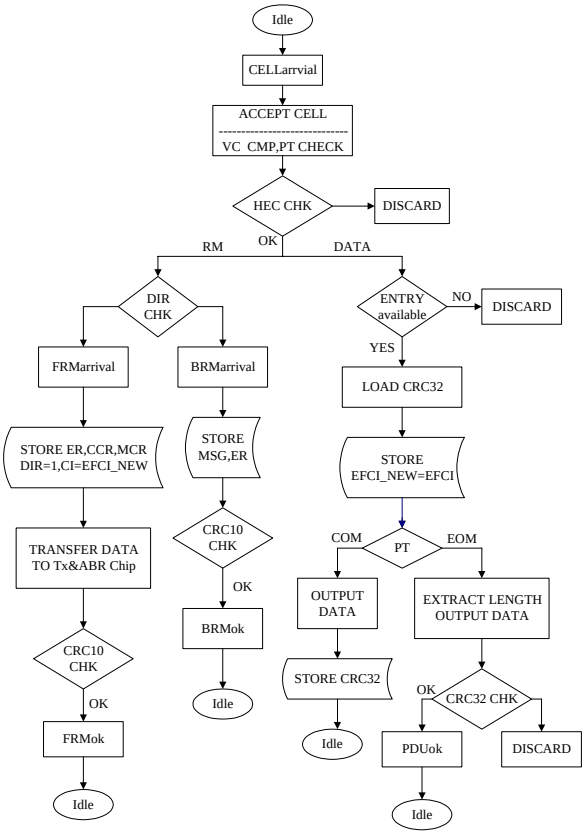


Fig. 7 Rx 晶片運作流程圖

— 如果是 Backward RM cell：

主機會經過中斷 (interrupt) 的方式被通知。除了 CRC10 欄位外，整個負載都會做 CRC10 的計算。Message，ER 等欄位會存起來，以待主機端讀取。在整個 cell 處理完後，CRC10 會被檢查，如果是正確的，主機會被中斷。

上述的運作流程如圖 7 所示。

C. ABR Controller 晶片運作流程

當主機啟動 ABR 服務時，ABR Controller 會依據主機端所寫入的資訊 (ACR，Nrm，Mrm，Trm) ，來決定發出何種的要求給 Tx 晶片；為了立刻知道網路的情況， ABR Controller 會先要求傳送 Forward RM cell，之後才是 Data cell 、Backward RM cell 等。

・傳送的邏輯如下

當到了該傳送的時間， ABR Controller 會先檢查 T_BUFFER 內有沒有資料要傳送或是 Rx 有沒有收到 Forward RM cell，若有，它會看看有多久沒傳 Forward RM cell 了，它的判斷依據如下：

— count ≥ Nrm：

Nrm 是在啟動時就下好的參數，表示傳 Nrm 個 data cell 後，必須摻雜一個 Forward RM cell，以確保 Forward RM cell 有傳的機會而不會被資料給淹沒。

— (count > Mrm) & (now ≥ last_RM + Trm)：

Trm、Mrm 亦是在啟動時就下好的參數，Trm 是表示多久的時間必須送一個 Forward RM cell，以提供傳送 Forward RM cell 時間時隔的上限。而 Mrm 是防止在速率降到很低的時候，只有傳送 Forward RM cell，因此必須要至少傳送了 Mrm 個 data cell，才允許其傳送 Forward RM cell，以合理的分配頻寬給 data cell 和 RM cell。

如果 ABR Controller 決定送 Forward RM cell，它會經由 Rx 晶片來中斷主機，主機便從 ABR Controller 讀出上次送 Forward RM cell 和現在的時間差。接下來就是由主機計算新的 ACR (Allowed Cell Rate)，算完後就將此值寫入 ABR Controller， ABR Controller 會立刻送出 Forward RM cell 的要求給 Tx 晶片。其中主機計算 ACR 的依據有二：

— (time > ADTF) & (ACR > ICR)：

如果太久沒送 Forward RM cell (time > ADTF) 而且 ACR 又大於 ICR (Initial Cell Rate) ，表示 ACR 的資料可能舊了，因此要

將速率降到 ICR。一般說，ADTF (ACR Decrease Time Factor)介於 0.01 ~ 10.23 秒之間。

— unack ≥ CRM：

如果有太多送出的 Forward RM cell 沒有轉為 Backward RM cell 回來，表示有太多 Forward RM cell 遺失了，應該把 ACR 降下來。一般來說， ACR 是以線性的方式減少，ACR = ACR – ACR * CDF (Cutoff Decrease Factor)，這段動作是在軟體部分執行。

如果 ABR Controller 決定不送 Forward RM cell ，它會選擇要送 data cell 還是 Backward RM cell ，它們有兩個判斷的標準，並且是 and 的關係：

— turn_around：

若 turn_around 為 true，則表示收到 Forward RM cell，且還沒將它轉為 Backward RM cell 送回去。

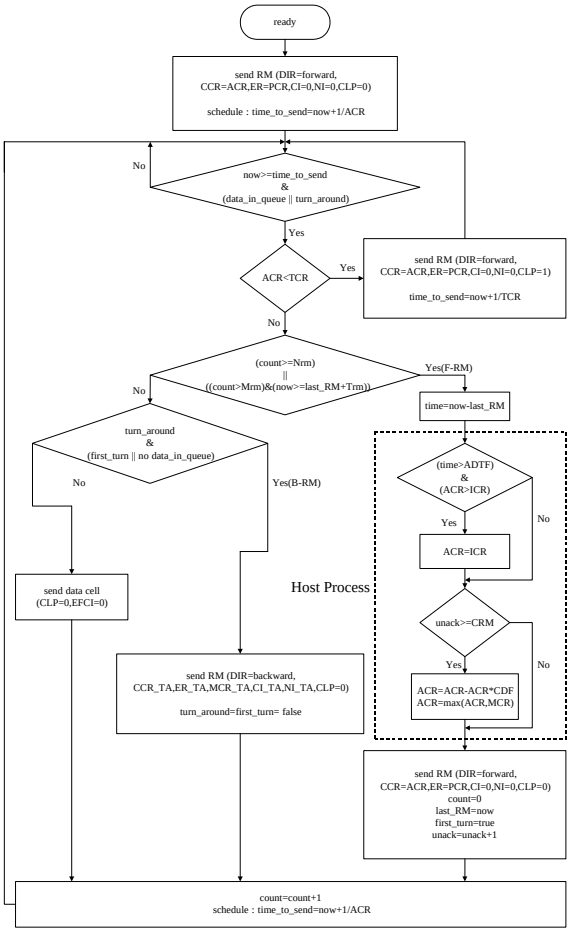


Fig. 8 ABR Controller 晶片傳送運作流程圖

— first_turn || no data_in_queue：

在上次送出 Forward RM cell 後，尚未傳送任何 Backward RM cell 之前 (此時 first_turn 為 true)，Backward RM cell 傳送的權值會比 data cell 高。而傳完一個 Backward RM cell 之後 data cell 的權值又會調回比 Backward RM cell 高。此舉是防止對方一直送 Forward RM cell 過來，而導致 data cell 的傳送頻寬減少。若收到 Forward RM cell (turn_around 為 true)，且沒有資料在等待傳送，則會直接傳送 Backward RM cell。

以上兩點都成立的話， ABR Controller 會送出 Backward RM cell 的請求給 Tx 晶片， Tx 會將先前 Rx 收到 Forward RM cell 而儲存在 Tx 的 7 個位元組 (BN， CI， NI， RA， ER， CCR， MCR)，組成一 Backward RM cell 傳給另一端。

否則， ABR Controller 會送出 data cell 的請求給 Tx ， Tx 依據此一請求從 T_BUFFER 取資料送出 data cell。注意，這裡是以 cell 為單位，而不是以封包為單位， Tx 收到請求後只會送一個 cell，而不是一個封包的長度。

上述的運作流程如圖 8 所示。

・接收的邏輯如下

當 Rx 晶片收到 cell 時，它會區別是哪一種 cell ，如果是 data cell，它會幫 ABR Controller 記錄最新的 EFCI 位元，此位元位在 data cell 標頭，當擁塞時，支援 ABR 機制的交換機會負責填此位元。

如果收到的是 Forward RM cell ， ABR Controller 就會被通知， ABR Controller 會將 turn_around 設為 true。

如果是收到 Backward RM cell，Rx 不會通知 ABR Controller，而是直接中斷主機，主機必須由 Rx 讀取 3 位元組的資料，包括 CI，NI，BN，ER 等資訊，來記算出最新的 ACR。

— 如果 CI 位元為 1：

表示網路擁塞，所以必須將傳送速率降下來，降多少由主機決定，一般是用線性的方式減少， ACR = ACR – ACR * RDF (Rate Decrease Factor)。

— 如果 NI 位元為 0：

表示網路無擁塞，所以主機可提昇傳送速率，提昇的多寡也是採用線性的方式， ACR = ACR + PCR * RIF (Rate Increase Factor)，這段動作是在軟體部分執行，但不能超過 PCR，ACR = min (ACR，PCR)。

— 如果 BN 位元為 0：

代表收到的 RM cell 是由自己所產生的，而不是由交換機或對方所產生的，因此可將 unack 歸零，表示自己送出的 Forward RM cell 有回來。若沒有回來，會在傳送邏輯被減速。

— ER：

如果交換機有支援 ER 的機制，如 ERICA (Explicit Rate Indication for Congestion Avoidance)，交換機會在 RM cell 的 ER 欄位，填入它可提供的速率給此一連線，主機便可清楚的將 ACR 調整到網路可乘載的最小速率， ACR = min (ACR，ER)。待主機算完後，立刻將新的 ACR 寫入 ABR Controller，此後， ABR Controller 就會將傳送的速率降到新的 ACR。

上述的運作流程如下圖所示。

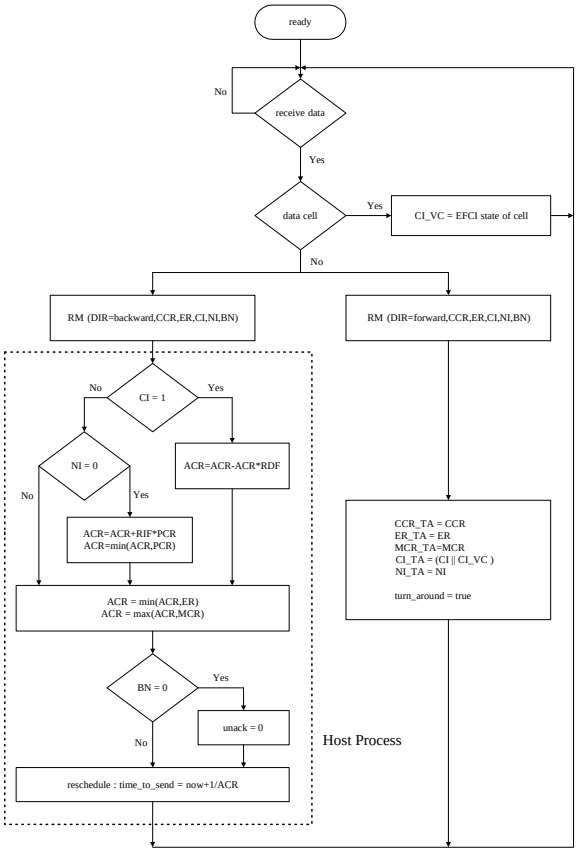


Fig. 9 ABR Controller 接收晶片運作流程圖

在了解我們的 Tx、Rx、ABR Controller 三顆晶片的運作流程及功能後，我們接下來看看，為了支援 ABR 機制，晶片設計上所遇到的一些問題與我們所使用的創新設計：

D. Tx 晶片設計的上的問題與創新

就 Tx 晶片而言，主要有下列幾項新的設計問題，大部份是關於 ABR 服務的。

・如何設計狀態機 (state machine) 來控制 Tx 晶片的運作流程？

因為對每個 ABR 連線來說，就有 4 種 cell，因此 Tx 必須處理 8 種 cell 才能支援兩個 ABR 連線。狀態機必須產生控制訊號來控制流程。有兩種方法來實作狀態機：Mealy 狀態機和 Moore 狀態機。 Mealy 狀態機的輸出除了跟現在的狀態有關以外，還跟輸入有關；而 Moore 狀態機只跟現在的狀態有關。

所以 Mealy 狀態機比較適合我們的設計。有限狀態機通常使用計數器 (counter) 來實作，一般來說有兩種計數器，Johnson 計數器和位元計數器，主要的差別如下所述：

— K 個正反器 (flip-flop) 在 Johnson 計數器能夠表示 2K 個狀態，而 K 個正反器在位元計數器中能夠表示 2K 個狀態。

— 在 Johnson 計數器，每個狀態需要兩個位元去解碼，在位元計數器則需 K 個位元解碼。

換句話說， Johnson 計數器的狀態比較容易解碼，但是卻需要較多的正反器，位元計數器則相反。在我們的設計中， Johnson 計數器和位元計數器在 FSM (Finite State Machine) 中合併著使用。Johnson 計數器是主要架構，而位元計數器則是應用在產生少數控制訊號的特殊狀態。

・如何實現資料管理 (data management)？

每個連線有 4 種 cell，對每種 cell 而言，被分為標頭和負載來處理。為了簡化記憶體的管理，上層的封包被暫存在外置的 FIFO (T_BUFFER)，RM cell 的 7 位元組負載會暫存在 Tx，而 ATM 的 4 位元組標頭則被各種 cell 所共享。

・如何使用新方法產生一個 cell？

因為要產生 8 種 cell ，因此需要設計 cell path 來結合標頭和負載來產生一完整的 cell。我們將 cell path 分為三個部份，第一部份是標頭路徑 (header path)，第二部份是 data cell 負載路徑 ，第三部份是 RM cell 負載路徑。

由 Main FSM 來控制， VPI/VCI 從暫存器中讀出。對 data cell 而言，負載由 T_BUFFER 中取得，封包長度和 CRC32 的值也從 Tx 的暫存器中

讀出。而對 RM cell 來說，負載和一些資訊也是從 Tx 的暫存器中讀出。其結構如下所示：

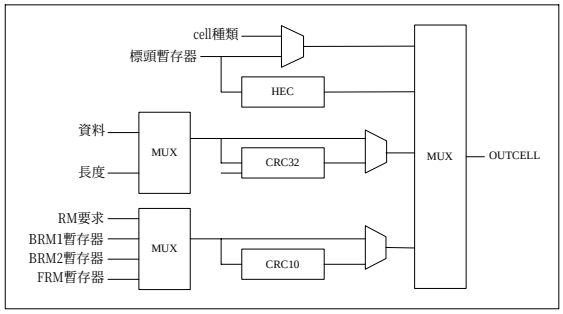


Fig.10 Cell Path 結構圖

・如何實作 CRC10 運算？

在 AAL5 和 ATM 協定中，只有 HEC (CRC8) 和 CRC32 的運算，但為了支援 ABR 機制，必須加上 CRC10 的計算。 HEC 和 CRC32 在之前所製作的網路卡中就已實作出來，在此不再贅述。在我們的設計裡，另外設計了一 8 位元平行處理的 CRC10 計算。

CRC10 的計算是由多項式產生器 (polynomial generator) $X^{10} + X^9 + X^5 + X^4 + X + 1$ 和 RM cell 的負載 (CRC10 本身所須的 10 個位元除外) 做模數 (modulo) 2 的運算。CRC10 編碼的電路結構如下圖所示：

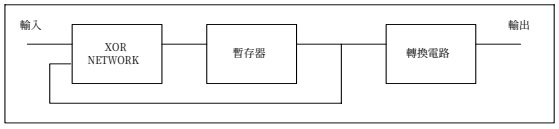


Fig.11 N-位元平行 CRC 編碼結構圖

因為 CRC10 碼是由模數 2 來記算，XOR NETWORK 能提供所須的運算, 而閉迴路 (closed-loop) 提供之前未算完的 CRC10 值，並和新的位元組做新的 CRC10 計算。

CRC10 在第 47 個 RM cell 負載位元組只有 6 個位元要加入運算，因此我們解決辦法是在最後補上兩個 0 的位元，當然，這樣做會影響實際的 CRC10 碼，所以設計了一個轉換電路將其修正回來。

・當 Rx 要覆寫 Backward RM cell 資料到 Tx 時，會遇到什麼問題以及如何使用創新方法解決？

如果 Tx 當時正在傳送 Backward RM cell, 我們必須禁止 Rx 在此時傳送新的 Backward RM cell 資料，才可以保護正在處理的 Backward RM cell 不會被破壞。所以我們設計了一個 BUSY 的訊號告知 Rx 目前是否有在傳送 Backward RM cell。只有在 BUSY 訊號降為 0 的時候，Rx 才允許寫入 Backward RM cell 資料至 Tx。如下圖所示：

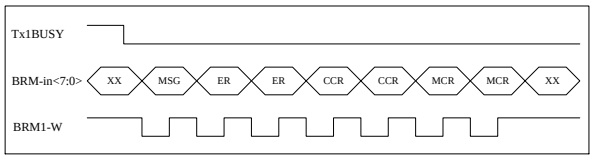


Fig.12 Tx 介面訊號時序圖

E. Rx 晶片設計上的問題與創新

・當收到新的 RM cell 時，舊的 RM cell 的處理方式？

當 RM cell 到達 Rx 時，Rx 會發訊號給 ABR Controller 或是主機。

如果是 ABR Controller 被通知，則 ABR Controller 在此時必須避免發送 Backward RM cell 的請求給 Tx，直到 Rx 將 Forward RM cell 資料的方向 (DIR 位元) 和擁塞 (CI 位元) 更新。然後覆寫到 Tx, 並確定 CRC10 的檢查無誤後, Rx 會再次通知 ABR Controller，表示允許 ABR Controller 發 Backward RM cell 的請求了。

若是主機被通知，表示收到的是 Backward RM cell，在 Rx 處理此 cell 的時候，必須避免主機在此時讀取 Backward RM cell 的資料，直到此 Backward RM cell 的 CRC10 檢查無誤後，主機才可讀取。這裡我們採用的解決辦法，是讓主機在此時讀出來的資料全為 0，表示連方向都不對 (Backward RM cell 的 DIR 位元為 1)，主機可以很輕易的區別此時讀出來的資料是不正確的。

・為支援兩個連線，如何有效的管理 R_BUFFER 來重組封包？

R_BUFFER 是由三個 banks 所組成，每個 bank 都可以被任意連線所使用，而它們是由一個 CAM (Content Addressable Memory) 所管理，而連線的標記 (index) 做為比對的依據。當收到 data cell 時，會去 CAM 檢查是否有還在等待重組的 bank，並且它是屬於哪個連線的，標記 0 或是標記 1，如此便能找到所屬的 bank。

CAM 的行位 (entry) 是由封包的第一個 cell 所註冊，在主機收完封包後會釋放掉。可參考在 Rx 運作流程中所介紹的 CAM Table。

・當 data cell 在 CAM 中找不到任何一個行位是空的時候，該用什麼方法解決？

當封包的第一個 cell 找不到行位註冊時，表示所有在 CAM 中的行位都在使用中，很自然的，這個 cell 不能被接受。但是我們必須注意到一件事，往後屬於同一封包的所有 cell 都可以丟棄了，因為封包將不是完整的。因此在我們的設計中，若是發生此情形，這些 data cell 都會被丟棄。

・如何維持 Rx 發出中斷 (interrupt) 的順序？

如果中斷的順序是錯的，那麼上層收到的封包順序將也會是錯的，也許上層會像 IP 協定一樣將之修正回來，但假使今天上層不是像 IP 這樣的協定，不去理會封包組完的順序似乎不是很好的做法。

因此，我們使用了一個創新的方法設計了一個類似矩陣的電路 — Matrix，去維持 CAM1，CAM2，CAM3 組完的順序，最早組完的擁有最高的權值，也就是唯一會發中斷給主機的 CAM，主機會依據這個資訊到正確的 bank 讀取封包資料。Matrix 的基本電路圖如下所示：

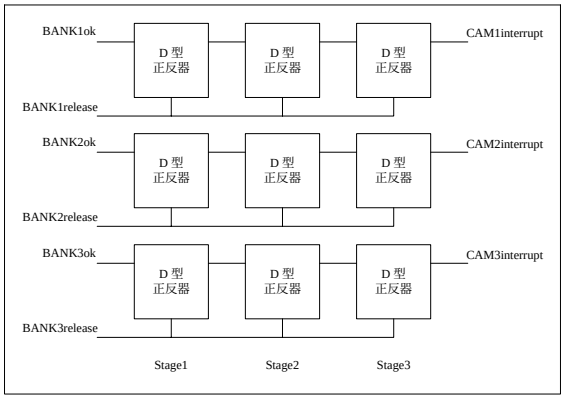


Fig.13 Matrix 的基本電路圖

圖中的 Stage1，Stage2，Stage3 代表不同的權值， Stage3 擁有最高的權值。當 bank 組完封包之後， BANK*n*ok 的訊號會進入 Matrix 中排隊，當其權值是最高時，就會透過 CAM*n*interrupt 去中斷主機。排隊的機制是以 D 型正反器來實現，在任一 BANK*n*ok 進入 Matrix 時， CAM 會發出訊號觸發 Matrix 中的所有正反器，若 BANK*n*ok 在 Stage1 則會跳到 Stage2，如果主機不去處理 CAM*n*interrupt，這樣的流程會一直到三個 Stage 都有訊號。當主機釋放 CAM 的時候，BANK*n*ok 也會被釋放。

F. ABR Controller 晶片設計上的問題與創新

・如何控制傳送的速率？

首先將基頻從原來的 12.5 MHZ 降到 48.828 KHZ (12.5M / 2^8)， ABR Controller 就以 48.828 KHZ 的時脈 (clock) 做為發送要求給 Tx 晶片的參考指標，因此對每條 ABR 連線而言，最高的傳送速率可達 48.828 Kcells / sec，換言之，將近 48.828 K * 8 * 48 = 18.75 M bits / sec (不含 ATM 標頭) 的傳送速率，對利用剩餘頻寬的 ABR 服務來說，已綽綽有餘，所以我們選擇此頻率為 ACR 的基頻。

至於如何調整 ACR，我們採用計數的方式，設計一 12 位元的計數器，此計數器可有 2^{12} (4096) 個參考值，例如，當主機寫入 10 到此計數器時，計數器會依時脈由 10 計數到 0，當計數到 0 時，ABR Controller 才會決定是否要傳送要求給 Tx 晶片，由於 ABR Controller 的時脈為 48.828 KHZ ，所以 ABR Controller 發送要求的速率會因此降至 4.8288 K cells / sec，降為最高傳送速率的 1 / 10 倍，而後計數器會再從 10 重新計數，如此便達到降低並控制傳送速率的功能。因為最多可計數到 4096，因此最小的傳送速率可達 11.92 cells / sec (48828 / 4096)。

利用計數來設計還有一個優點，那就是速率降低是以 1/N 降低的方式 (PCR / 1，PCR / 2，PCR / 3，……)，而非線性的。雖然在較高的速率下，ACR 下降的精確度會較線性方式來得差，但是若是在較低的速率下，速率反而能有較高的解析度。而在真實的環境中，我們會期望若發生擁塞的情形時，擁有高頻寬的使用者能降低較多的速率，而低流量的用戶只降低少許速率，才能達到所謂的公平性，只要不低於所保證的最小速率 (Minimum Cell Rate, MCR) 就行了。

所以我們在設計網路卡時，決定採用 1/N 速率降低的方式，在高速時降幅較大，而在越是低的速率下，越能提供準確的解析度，以符合實際所需。我們從下圖可清楚的看見 ACR 解析度變化的情形：

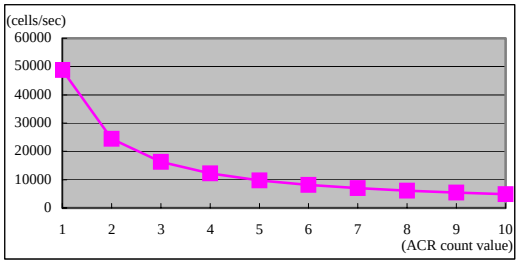


Fig.14 ACR 解析度變化

・如何計算上次送出 Forward RM cell 到現在的時間 (time = now - last_RM)？

為解決這個問題，我們設計了另一個計數器，負責計數時間。當送出 Forward RM cell 的要求時，計數器會歸 0，計數器的時脈 (clock) 每跳一次, 計數器就會加 1, 直到送出 Forward RM cell 的要求，又會歸 0。

此計數器的基頻設計在 1525.88 HZ (12.5M / 2^{13})，表示每 0.655 msec 就會計數一次，以 14 位元計數器來設計，時間可計算到 10.73 sec (0.655 * 2^{14}) 。範圍由 0.655 msec 到 10.73 sec ，比 ABR 規格所要求的 10 msec 到 10.23 sec 之範圍要來的大；而解析度 0.655 msec 也比規格所要求的 10 msec 要來的精確。

・當要送出 Forward RM cell 時，如何通知主機？

當要送出 Forward RM cell 時, 需要一些計算，我們的設計方式是 ABR Controller 透過 Rx 去中斷主機，主機可利用 Rx 內的中斷狀態描述子 (interrupt status descriptor)去了解中斷的來源。

INTmask	ABR2	ABR1	BRM2	BRM1	CAM3INT	CAM2INT	CAM1INT

Fig. 15 中斷狀態描述子

CAM1INT 表示中斷是 CAM1 所發的；CAM2INT 表示中斷是CAM2 所發的；CAM3INT 表示中斷是 CAM3 所發的；BRM1 表示中斷是由於收到第一個連線的 Backward RM cell ； BRM2 表示中斷是由於收到第二個連線的 Backward RM cell ； ABR1 和 ABR2 表示中斷是由 ABR Controller 分別為連線 1 和 2 的 Forward RM cell 所發的。

・有關 ACR 的浮點運算部分。

由於 ABR 服務需要乘除法的計算，這勢必會增加許多硬體成本，因此必須評估硬體化後的效率。以 MCS-51 系列的 8 位元單晶片為例，其工作頻率為 24 MHZ ，所需的指令週期都是以 8051 指令週期為準。經大略估計 8051 單晶片中一些四則運算所需 machine cycle 如下所示:

- Floating Point +Floating Point：66 Machine Cycle
- Integer * Floating Point： 106 Machine Cycle
- Constant / Floating Point： 87 Machine Cycle

	送 Forward RM (cycle)
Floating Point + Floating Point	2*66
Integer * Floating Point	2*106
Constant / Floating Point	1*87
Total	431

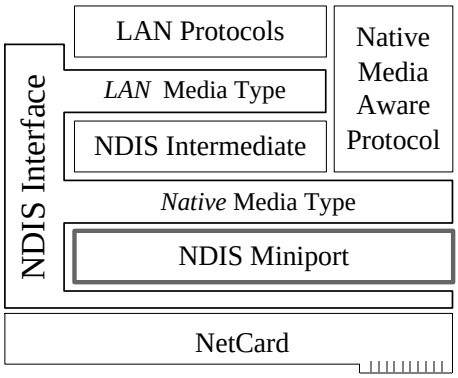
就送 Forward RM cell 而言，估計須 431 個 machine cycle ，其中還不包含一些整數的比較，將 ACR 結果轉成實際上的格式以及載入的時間。

因為一個 machine cycle 為 12 個 clocks 且單晶片的頻率 24 MHZ ，所以傳送 Forward RM cell 最少要花 216 μsec 來計算，若以 10 Mbps 的連線速率來看，計算一次 ABR 結束，到調整好 ACR，最少已有 5 個 cell 可流入網路中。

所以就這顆單晶片而言，硬體化似乎不可行，因為若我們以透過 PCI 的介面將 ACR 的運算交由主機去處理，顯然並不會比我們將 ABR 硬體化來得慢，那我們將它硬體化就沒有意義了，而且我們的 ACR 的運算經過了相當的簡化，所以我們選擇將其交由主機來運算。而經過實際測量，交由主機運算到調整好 ACR，也僅須 20 μsec (Intel Celeron 500 CPU)。

(二)、ATM網路卡的軟體架構

我們在 Microsoft Windows 95/98/NT 上的 NDIS Miniport Driver 皆是以符合 NDIS 4.0 的規格寫成，程式在作業系統中扮演的角色以及作用如下圖所示：



在這裡不再多提有關 NDIS 的系統呼叫 (System Call)，而只討論有關我們的網路介面部分的程式。

A. 傳送端的資料與控制流程

見圖16，當主機上有資料要透過網路傳送時，它會發出軟體中斷通知驅動程式，而驅動程式會先對 Tx 晶片作設定，包括這筆封包的長度，相對應之 VPI/VCI，而後檢查 Tx 晶片的狀態。

這個狀態會告訴程式 Tx 晶片是否在忙碌中，如果是，資料則不能往硬體丟出。這個忙碌訊號代表 Tx 晶片還沒將資料處理完, 資料處理速度是由 ABR 晶片為了控制cell的速度來決定。

當主機設定完 Tx 晶片後，將欲傳送的 CPCS-PDU 搬到 Transmission Buffer (Tx_Buffer)。接下就是Tx 晶片的處理動作，這些已在上節討論。

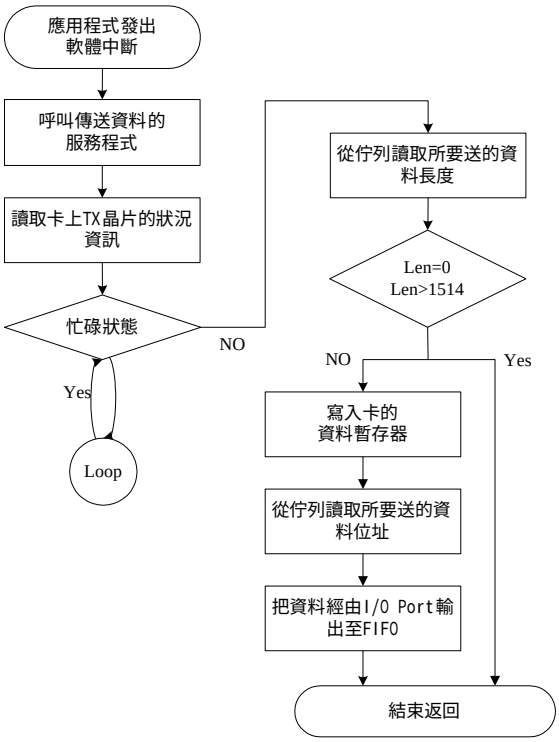


Fig. 16 驅動程式傳送資料流程

B. 接收端的資料與控制流程

當主機端的驅動程式收到 Rx 晶片發出的中斷 (Interrupt) 時，它會藉由讀取 Rx 提供的狀態暫存器 (Status Register) 得知這個中斷的種類，來決定做相對應的動作。

這些種類包括：

・資料到達

分為三種狀態，代表編號1~3的Bank有資料進來。

・Forward RM cell 發出

是在 ABR controller 要發出 Forward RM cell 時通知驅動程式，要求驅動程式依照演算法算出 ACR 值，並製造出 Forward RM cell 的內容 (payload) 再寫入暫存器中。

・接收 Backward RM cell

當硬體接收到 Backward RM cell 時會通知驅動程式，驅動程式則從暫存器中讀出新的ACR值，代入演算法中做更新。

上述的運作流程如圖17所示。

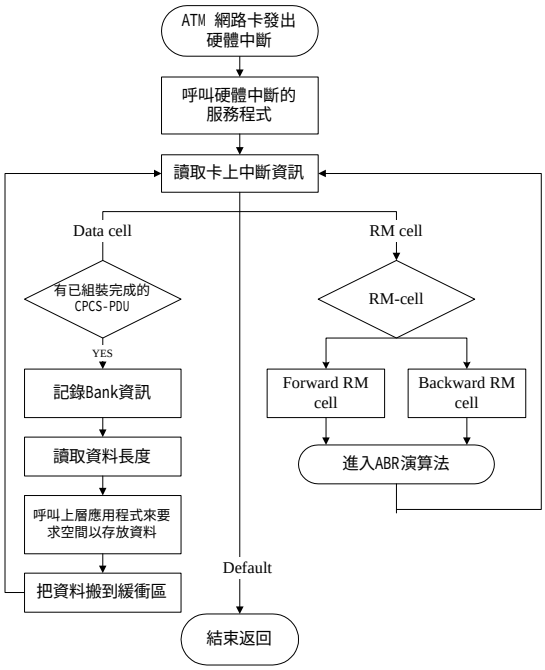


Fig.17 驅動程式接收資料流程

C. 軟體中的 ABR 演算法

從硬體中我們可以知道，從ACR暫存器讀出來的值其實是可利用細胞速率 (Available Cell Rate，ACR) 的倒數，1/ACR，而演算法中最主要的除法就是把 1/ACR 換成 ACR 值，等到算完後再將 ACR 轉換為 1/ACR 寫入暫存器，但是這樣的作法需要浮點運算，也沒有必要。

因為我們從之前硬體的描述可以知道，ABR controller 是以除頻來達到減低速率的要求，它所需要的ACR值其實就是除頻算式的除數，而且是一個整數，也就是即使算出一個浮點數也要轉換成整數才能拿來除頻。

而且經由前面的硬體設計描述，我們可以知道給的除數值小，硬體調整速度的解析度會較小，速度降得較快；給的除數值大，也就是在傳送速度較低的時候，解析度自然會提升。軟體中線性的演算此時顯得沒有意義，所以我們可以利用硬體這樣的特殊設計來更改演算法，使得演算法中所記錄的ACR值就代表了 1/ACR。

當需要降速時，ACR 的值加一，代表除數增加最小單位一；當需要增速時，ACR 的值減一，代表除數減少最小單位一。而實際傳送速度下降的幅度也會自動的依照 ACR 值的大小有所改變。

所以依照此邏輯來推算，演算法中的變數如：峰值細胞速率 (Peak Cell Rate，PCR)，初始細胞速率 (Initial Cell Rate，ICR)，最小細胞速率

(Minimum Cell Rate，MCR)，因為是一開始就約定好的，不需要變動，所以我們在填值的時候也依照修改 ACR 的方法，給定配合硬體的值。

而且演算法的算式都要反過來，如：取最大值變成取最小值，取最小值變成取最大值等等。

最後，軟體內的演算法流程可以用下面兩個圖來表示：

・Forward RM cell

從圖18中，我們可以看到由於 Forward RM cell 演算法的改變，流程中只有兩個加減運算，三個判斷式, 以及九個 I/O 讀寫動作, 在這裡面 CPU 時脈越快，演算法也越快，但是最後的瓶頸會落在 I/O 的快慢。

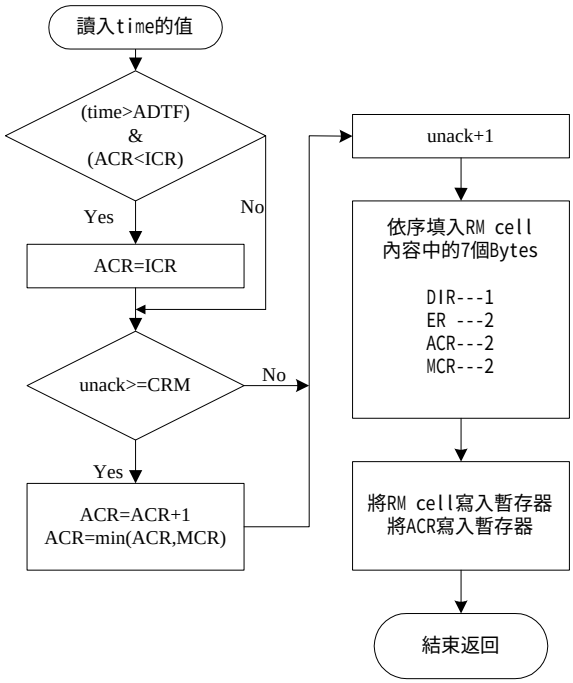


Fig.18 Forward RM cell 演算法

・Backward RM cell

我們從ABR晶片上的暫存器中讀出 Backward RM cell 的內容 (Payload)，Octet7~9 記載演算法所需要的資訊，其中Octet7的內容如下圖：

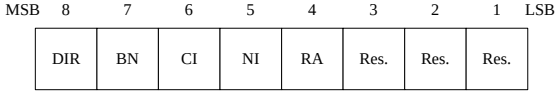


Fig.19 Message Type Field of Octet 7

DIR = 0 表示 Forward RM Cell，
= 1 表示 Backward RM Cell。
BN = 0 表示這不是發 RM Cell 的來源端所發出的，它可能是交換機或目的端所發出的。
CI = 0 表示沒有壅塞，
= 1 表示有壅塞產生。
NI = 1 表示不允許增加速度，
= 0 表示可以增加速度。
而其它的位元我們並沒有使用到。

Octet 8 和 Octet 9 代表明確速率值 (Explicit Rate，ER) 的高位元和低位元，有了這些值，我們代入圖20的演算法中。

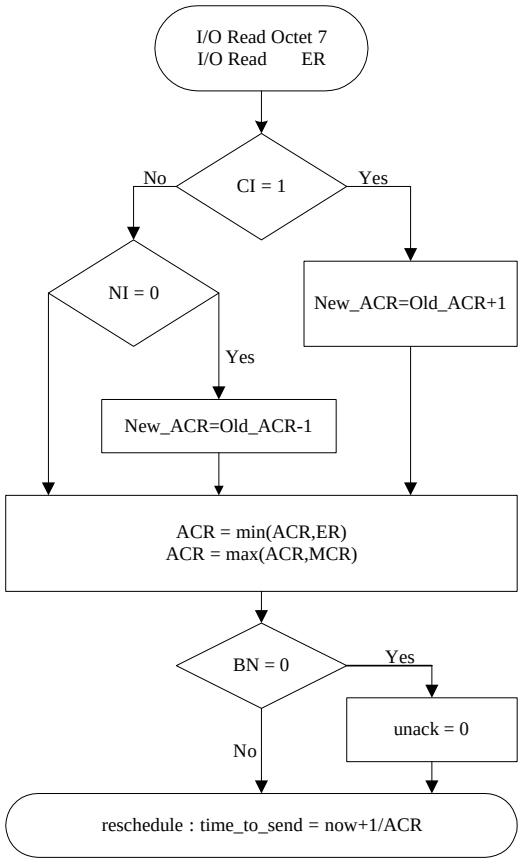


Fig. 20 Backward RM cell 演算法

在流程圖中我們可以看到，一次的完整的運算只會有一個加減，四個比較，與五個 I/O 動作，同樣的，當 CPU 的速度越來越快，演算也會加快，最後的瓶頸也是落在 I/O 的快慢。

五、實驗結果與比較

(一)、最小細胞速率 (MCR) 的保證

MCR 的保證有賴 ATM 交換機 (switch) 和 ABR 主機 (host) 的相互搭配。在 ATM 交換機有能力保證 MCR 的條件下, 主機只需確定計算出來的 ACR 一定不低於 MCR，即可保證端對端 (end-to-end) 的最低速率 MCR。

這一部份的計算，我們實現在驅動程式上面，

我們驗證驅動程式運作正確無誤，主機端的 MCR 保證也就達成了。

(二)、訊務整流 (Traffic Shaping) 的效能

為了要評估訊務整流的效能，我們以不同的傳送速率 (ACR) 來觀測整流的情形，協定分析儀會記錄屬於該連線所有細胞的到時間及屬性 (如RM cell等)，其精準度可達 100 ns。 ATM 交換機從 TAXI的輸入埠 (100 M bits / sec) 將細胞送往 OC-3 的輸出埠 (155 M bits / sec) ，所以在實體層會有 Jitter 的現象發生。另外，我們將 Nrm 設為 12 ，表示每送出 12 個 data cell 時, 會送一個 Forward RM cell ，這些 overhead 都算在我們統計的數據之中。

・測試環境 ：

— Nrm = 12，Nrm 是在啟動時就下好的參數，表示傳 Nrm 個 data cell後，必須傳一個 Forward RM cell，以確保 Forward RM cell有傳的機會而不會被資料給淹沒。

— Mrm = 2，Mrm 是防止在速率降到很低的時候，只有傳送 Forward RM cell，因此必須要至少傳送了 Mrm 個 data cell，才允許其傳送 Forward RM cell，以合理的分配頻寬給 data cell 和 RM cell。

— Trm = 7.86 (msec) ，Trm 是表示多久的時間必須送一個 Forward RM cell，以提供傳送 Forward RM cell 時間時隔的上限。

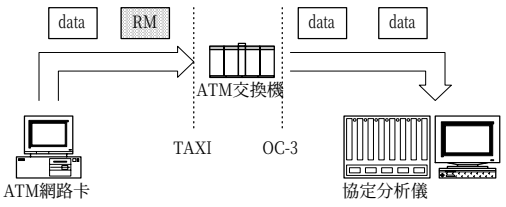


Fig. 21 展示平台示意圖

我們以不同的傳送速率 (ACR)，來分析訊務整流的效能。

・理論值 ：

ACR (cells/sec)	ACR (bits/sec)	傳送細胞間的時間間隔(ns)
24414	9,375,000	40960
12207	4,687,500	81920
8138	3,125,000	122880
4882	1,875,000	204800
3051	1,171,803	327680
1525	585,937	655360

・實驗值 ：

(a) ACR = 24414 (cells/sec)

cell到達的時間(sec)	到達時間間隔(ns)
36.6545981	
36.6546346	36500
36.654677	42400
36.6547194	42400
36.6547592	39800
36.6548016	42400
…	…

平均值： 40961.76 (ns)
標準差： 5963.06 (ns)
標準差百分比： 14.55 %

(b) ACR = 1220 (cells/sec)

cell到達的時間(sec)	到達時間間隔(ns)
43.425511	
43.4255959	84900
43.4256753	79400
43.4257575	82200
43.425862	104500(RM)
43.4259414	79400
…	…

平均值： 83097.05 (ns)
標準差： 8443.16 (ns)
標準差百分比： 10.16 %

(c) ACR = 8138 (cells/sec)

cell到達的時間(sec)	到時時間間隔(ns)
9.536925	
9.5370469	121900
9.5371715	124600
9.5372933	121800
9.5374375	144200(RM)
9.5375594	121900
…	…

平均值： 123820.58 (ns)
標準差： 8940.06 (ns)
標準差百分比： 7.22 %

(d) ACR = 4882 (cells/sec)

cell到達的時間(sec)	到達時間間隔(ns)
29.3866939	
29.3868974	203500
29.3871014	204000
29.3873081	206700
29.3875344	226300(RM)
29.3877357	201300
…	…

平均值： 205332.35 (ns)
標準差： 8341.23 (ns)
標準差百分比： 4.06 %

(e) ACR = 3051 (cells/sec)

cell到達的時間(sec)	到達時間間隔(ns)
57.3653637	
57.3656923	328600
57.3660204	328100
57.3663713	350900(RM)
57.3666945	323200
57.3670226	328100
…	…

平均值： 328255.88 (ns)
標準差： 8285.92 (ns)
標準差百分比： 2.52 %

(f) ACR = 1525 (cells/sec)

cell到達的時間(sec)	到達時間間隔(ns)
31.9509489	
31.9516056	656700
31.9522601	654500
31.952914	653900
31.9535707	656700
31.9542247	654000
…	…

平均值： 656576.47 (ns)
標準差： 5121.91 (ns)
標準差百分比： 0.78 %

・理論值與平均值的誤差及誤差百分比 ：

ACR (cells/sec)	理論值	平均值	平均誤差 (ns)	誤差百分比 (%)
24414	40960	40961.76	1.76	0.0043
12207	81920	83097.05	1177.05	1.4368
8138	122880	123820.58	940.58	0.7654
4882	204800	205332.35	532.35	0.2599
3051	327680	328255.88	575.88	0.1757
1525	655360	656576.47	1216.47	0.1856
平均			685.03	0.4121

・結果分析 ：

當傳送速率在 10 M bits / sec 左右的時候，細胞經 ATM 交換機到協定分析儀的時間間隔有較大的變異度，其中一部份原因是由於網路卡將ACR之運算交由主機處理的緣故 (大約 20 µsec 的處理時間)，而另外一部份則是由於ATM 交換機傳送延遲的關係以及 TAXI 介面與 OC3 介面 速率的差異。

而當傳送速率在 1 M bits / sec 左右的時候，時間間隔的變異度很明顯的由原來的 10 % 降到2%，除了ATM 交換機延遲的影響減少了以外，對約400 µsec 的傳送時間間隔而言，主機處理的時間算是相當的少，如果 Nrm 能再降低，則整體的變異度相信能再降低。

(三) 、網路擁塞情況時的表現

為了測試網路卡的實用性，我們將網路卡放置視窗98的環境中，並刻意由協定分析儀製造擁塞，以觀察網路卡實際效率。

・測試環境 ：

- ICR = 6975 (cells/sec) = 2.678 (Mbits/sec)
- PCR = 6975 (cells/sec) = 2.678 (Mbits/sec)
- MCR = 4096 (cells/sec) = 1.573 (Mbits/sec)
- Nrm = 32
- Mrm = 2
- Trm = 32.75 (msec)
- CRM = 4
- ADTF = 167 (msec)
- 協定分析儀的交通量 ：
  協定分析儀所產生的交通量為一突發的CBR (Constant Bit Rate)，大約維持數秒鐘，隨後便解除擁塞。在此次實驗中，我們設定 CBR = 149 Mbits/sec，並在協定分析儀觀察傳送端的傳送速率的變化情形。

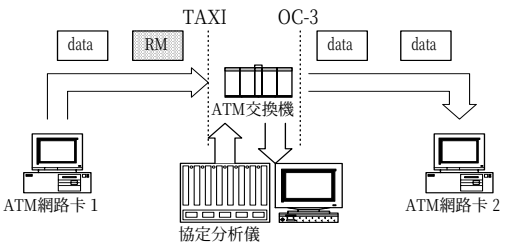


Fig. 22 測試平台示意圖

・實驗結果:
如圖 23 所示。

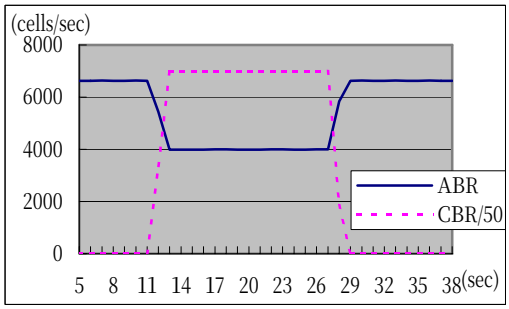


Fig. 23 Windows 98 下的實驗結果

協定分析儀是以每秒來統計的，不過仍然可看出我們的網路卡能即時的調整速率，當網路進入擁塞狀態時能降低傳送速率，當網路回到無擁塞狀態時亦能迅速地提高傳送速率；更重要的是，它同時保證了最小細胞速率 (MCR = 4096 cells/sec)。

(四) 、對外的互通

我們將卡連到 FORE ASX-200 ATM 交換機，並透過交換機連上中正大學區域寬頻網路接取中心 (GigaPOP)，再連到國家寬頻實驗網路 (NBEN)，到達國家高速電腦中心，最後回到我們的另一台主機，這樣設定的目的，是希望我們主機送出的資料能真正的在廣域網路中運作。而實驗結果也證明我們 ATM 網路卡所送出的細胞，能順利的經由對外的交通後，返回另一張 ATM 網路卡。在此我們並不分析 ABR 的效能，因為對外網路的環境並不是都支援 ABR 機制。

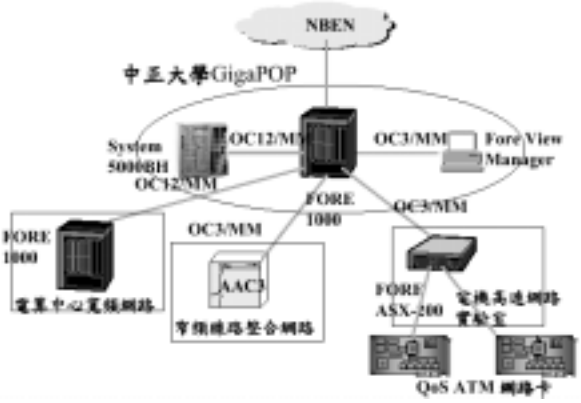


Fig. 24 廣域網路連線圖

六、結論

我們在這次專題中實作出了具 QoS 功能的主機端網路介面。它具有隨網路壅塞狀況而機動調整其速率的功能，以及提供訊務整流 (Traffic Shaping) 的能力，並且能支援 IP over ATM 協定，與區域寬頻網路接取中心 (GigaPOP) 和國家寬頻實驗網路 (NBEN) 中的 ATM switch 互通。

在創新方面，它的 FPGA 晶片使用了新的設計方式，如：狀態機 (state machine) 設計、傳送速率的控制，這是設計方法創新；專題中的 ABR 服務程式中沒有浮點數的出現，增快了演算法運算的速度，也減低了所需佔用的資源，這也是設計方法創新；ABR 服務協定使用新的觀念，如：配合硬體創造出新的運算式，這是理論創新。

在應用方面，本專題可應用的範圍非常的廣泛，從視訊會議 (Video Conference)、遠距教學 (Distance Learning)、以及隨選視訊 (Video on Demand)、到遠距醫療服務，我們的專題成品皆可以有良好的表現，以及絕佳的品質。在 採用 ATM 為骨幹傳輸技術的網際網路已經逐步成為商業主要的傳輸服務網路的同時，我們的作品提供了網路服務業者一個良好主機端產品的雛形。

在完整度上，我們除了研發出完整的硬體與 WINDOWS 95/98/NT 的驅動程式與介面，還研發了 FreeBSD 等其他作業系統的驅動程式，並與 TCP/IP 協定一起做了完整的測試。而且專題成品也經過 Fore ATM switch 與 ATM 協定分析儀各種流量狀況的功能與穩定測試。

這些條件使我們專題成品的創新度，穩定性，效率，以及相容性皆有非常優越的表現。

隨著 Internet 的快速發展，通訊廠商發現至少有兩個重要的趨勢推動寬頻網路服務：Internet 將成為電子商務的交易網路；以及一系列整合連線設備 (Integrated Access Device) 推陳出新，未來利用 Internet 傳輸大量的語音、文字與影像的資料，並成為商業傳輸服務網路不再是夢想，這時網路服務品質的保證將變成網路服務業者 (ISP) 最重要的課題。網路端結合 IP 與 ATM，與主機端的 QoS 網路介面勢必成為未來市場寵兒。

七、參考文獻

[1] W. R. Stevens， “TCP/IP illustrated, Volume 1,” Addision-Wesley，1994

[2] ATM Forum. “ATM User-Network Interface3.1 Specification”，af-uni-0010.002，1994

[3] IETF， “Classical IP and ARP over ATM”，RFC 2225，April 1998

[4] IETF，“Requirements for Traffic Engineering Over MPLS”，RFC 2702，September 1999

[5] ITU，”Video conferencing on ATM”，H.321，Feb. 1998

[6] IETF，” RSVP Operation Over IP Tunnels”，RFC2746，January 2000

[7] IETF，”RSVP Diagnostic Messages”，RFC2745，January 2000

[8] ITU-T. “B-ISDN ATM Adaptation Layer (AAL) Specification.”，I.363，March 1993.

[9] Actel Corporation. “Actel FPGA data book.”，1999

[10] Advanced Micro Devices，Inc. “AM7968/AM 7969 - 125 TAXI chip Integrated Circuits Data Sheet and Technical Manual.”

[11] ATM Forum. “ATM Traffic Management 4.1 Specification.”，af-tm-0121.000，March 1999

[12] Microsoft Device Driver Kit，NDIS 4.0

[13] K. J. Chen and C. L. Chang “Universal Parallel CRC Circuit and Algorithm.”，R.O.C. patent.

[14] C. L. Chang， T. C. Hou， Y. S. Chu， and K. J. Chen. Throughput Characterization of a PC with a High Speed ATM Network Interface. In Pro. IEEE Globecom’96 Workshop on Transport Protocols for High-Speed Broadband Networks， Nov. 1996.

[15] MINDSHARE，“PCI system architecture.” Addison Wesley，4/E 1999