行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告 □期中進度報告

寬頻電信等級開放平台建置計畫

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 94－2219－E－009－023－
執行期間：　94 年　11 月　01 日至　　95 年　10 月　31 日

計畫主持人：張仲儒
共同主持人：王國禎、周勝鄰、陳健及簡榮宏（依姓名筆劃順序）
計畫參與人員： 黃嘉淵、徐彬海、胡洛川、曾恕康、賴建國及劉彥良等。

成果報告類型(依經費核定清單規定繳交)：□精簡報告　■完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
■出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
■涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：交通大學

中　華　民　國　95　年　12　月　31　日

中文摘要：

電信等級的網路設備向來以私有(Proprietary)架構為主，但在降低開發時程與成本的需求下，目前也開始趨向標準化的開放式架構。結合 ATCA (Advanced Telecom Computing Architecture)、Carrier Grade Linux 與 Open Specifications for Service Availability 的開放式架構，將會是未來通訊產業的重要趨勢之一。目前日本 NTT DoCoMo、韓國 KT 等網路服務業者已開始採用 ATCA 作為各種應用服務與網路基礎建設的共通平台。在 Carrier Grade Linux 方面則有 MontaVista、RedHat、WindRiver 等軟體廠商與其他 OSDL (Open Source Development Lab.)成員投入 Linux 的改良。在 Open Specifications for Service Availability 方面則有 SAF (Service Availability Forum)定義了相關介面規格，來介接底層作業系統與硬體的備援設計和上層應用服務軟體程式及系統整體管理功能。

我國通訊產業向來以硬體為主，軟體開發能力相對較弱，對於提升產品高附加價值及可靠度方面，一直有待加強，而標準化的開放式架構的趨勢提供了國內產業一個切入高階市場的機會。本建置計畫主要將結合產學研界，共同引入符合國際技術趨勢之"寬頻電信等級開放平台架構"(CGOA, Carrier Grade Open Architecture)，採用 ATCA Chassis 硬體架構及 Carrier-Grade Linux 作業系統，以及符合 SAF 標準之 High Availability middle-ware，共同以 EPON(Ethernet Passive Optical Network)系統為載具，在產業界所採用的硬體平台與軟體架構下，由學界進行架構設計改良與效能分析模擬，並建置 EPON 可靠度測試環境之實驗平台。預期經由本建置計畫的執行，將可有效提升電信設備系統之可靠度，加速我國電信等級設備之品質提昇，同時促使新技術與新應用服務的快速發展。

關鍵詞：Carrier Grade、Broadband、Service Availability、EPON、Open Architecture

英文摘要：

In opposition to conventional proprietary architectures, the standardizing open architectures have been adopted in network industry for reducing the time and cost of development of carrier grade equipments and systems. Open architecture is the new trend of communication industry that integrates ATCA (Advanced Telecom Computing Architecture), Carrier Grade Linux, and Open Specifications for Service Availability. Some service providers such as NTT DoCoMo Japan and KT Corp. South Korea, have used ATCA systems as common platforms for networking and services provision. Sofeware venders like MontaVista, RedHat, WindRiver and so forth, are engaged in development of Carrier Grade Linux. In addition, the SAF (Service Availability Forum) defines Open Specifications for Service Availability that are interfaces to high available hardware platforms in underlayer and application services including management in upper layer.

The Taiwaness communication vendors usually focus on hardware and are weak relatively for development of software and high reliable products with high add-on value. It is a obvious opportunity to enter the high level product markets for vedors in Taiwan by the open platform technologies. For following the recent tendency towarding open architectures, this project introduces the CGOA (Carrier Grade Open Architecture) technologies by gathering industrial, academic, and research teams. It builds EPON (Ethernet Passive Optical Network) system on an ATCA-based Chassis open architecture platform including Carrier-Grade Linux and SAF-compliant High Availability middleware. This system offers high reliable EPON trial environment on the hardware platform and software framework that was supplied by industry team and developed with technogies by academic team. It is looking forwarding to fruitful execution results that foster the carrier grade telecom system development with high reliability and quality, and encourage the evolution of new network technologies and services.

Key Words：Carrier Grade, Broadband, Service Availability, EPON, Open Architecture

目錄：

第1章、 前言 — 計畫背景說明

隨著各種嶄新多元的網路應用及服務的蓬勃發展，高速、穩定可靠，而且能連續不中斷的網路環境與服務，已是企業及個人客戶不可缺少的基本需求。因此，未來的各種網路應用服務的落實，勢必要仰賴一個穩定(predictable)、可靠(reliable)的網路基礎架構，不論是網路電話(VoIP)、視訊會議、多媒體娛樂、企業電子資料交換，都必須倚賴可靠的連結，以及穩定可預測的效能。因此，在整個網路基礎架構的組成元件中，包括網路線(如：光纖、銅纜)、電信設備(如：交換器、路由器)、管理系統(如：設定管理軟體、頻寬管理軟體)等的可用度(availability)如何提升，甚至達到電信等級的 99.999%高可用度要求，是網路、設備及服務提供者的最大挑戰。

傳統電信等級的網路設備或伺服器向來以私有(proprietary)架構為主，但在降低開發時程與成本的需求下，目前也開始趨向標準化的開放式架構。開放式架構涵蓋了硬體機箱、作業系統、可靠度架構三方面。在硬體機箱方面，PICMG(PCI Industrial Computer Manufacturers Group)所定義的 Compact PCI 標準(PICMG 2.X)目前已為許多 VoIP 相關產品所採用，但在內部通訊頻寬與電源及散熱方面有其限制。新一代的 ATCA (Advanced Telecommunication Computing Architecture)標準(PICMG 3.X)採用了最新的高速內部通訊介面，包括了 Gigabit Ethernet/10G Ethernet、FiberChannel、RapidIO、PCI Express，提供了充足的內部通訊頻寬，電源與散熱方面也預留了較大的發展空間，因此目前已有許多廠商積極採用。在作業系統方面，隨著 Linux 社群的快速擴展，越來越多的網路設備採用 Linux 作業系統及其搭配的各式網路協定與應用程式，來縮短開發時程與增加產品的功能性。但目前 Linux 的應用仍侷限在低階的個人/家庭網路設備上，電信等級的設備仍以私有架構、特製化的作業系統為主流，主因在於 Linux 本身以個人電腦應用出發，在電信等級可靠度的發展起步較晚。目前 OSDL(Open Source Development Labs)已成立了 Carrier Grade Linux 的 work group，針對現有 Linux 作業系統的 performance、clustering、security、serviceability、availability 等方面進行改良，提升到電信等級。在可靠度架構方面，則有 SAF (Service Availability Forum)定義了中介軟體(middleware)的功能需求，以及向上介接應用服務軟體程式、向下介接底層作業系統與硬體的備援設計、向外提供系統整體管理功能等各相關介面的規格，作為各種應用服務、作業系統、硬體平台支援電信等級可靠度的共同介面標準。

標準化的開放架構將會是未來通訊產業的重要趨勢之一。目前日本 NTT DoCoMo、韓國 KT 等網路服務業者已開始採用 ATCA，作為各種應用服務與網路基礎建設的共通平台。在 Carrier Grade Linux 方面則有 MontaVista、RedHat、WindRiver 等軟體廠商與其他 OSDL (Open Source Development Lab.)投入開發相關產品與技術。在電信等級可靠度方面，則有 SA Forum 成員的 GoAhead、Clovis 等公司進行 middleware 相關產品與技術的研發，而 HP、IBM、SUN、Fujitsu Siemens、Motorola、Ericsson 等公司進行電信等級系統設備的研發。

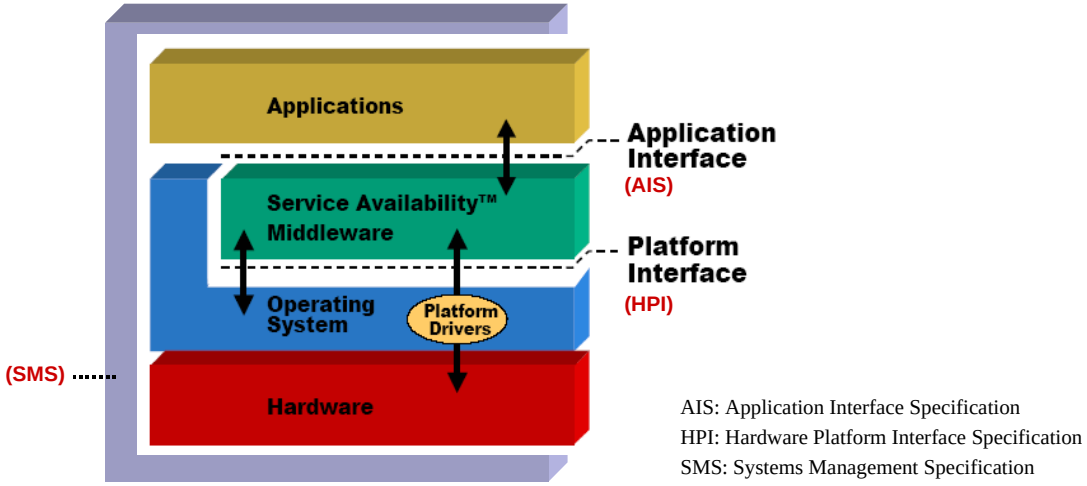
我國通訊產業向來以硬體為主，軟體開發能力相對較弱，對於提升產品高附加價值及可靠度方面，一直有待加強，而標準化的開放式架構的趨勢提供了國內產業一個切入高階市場的機會。學界以往在可靠度架構研究與效能分析多有著墨，但多半侷限於假想的模式與理論分析，與實際產品狀況與需求仍有落差。此次所規劃之「寬頻電信等級開放平台建置計畫」主要將結合產學研能量，共同引入符合國際技術趨勢之標準化的開放架構，採用 ATCA Chassis 硬體架構及 Carrier-Grade Linux 作業系統，以及符合 SAF 標準之 High Availability middle-ware。學界將與產業界與研究單位(如：工研院電通所、中華電信研究所)共同以 EPON(Ethernet Passive Optical Network)系統為載具，在產業界所採用的硬體平台與軟體架構下，進行架構設計改良與效能分析模擬，並將建置 EPON 可靠度測試環境之實驗平台。預期經由本建置計畫的執行，將可有效提升電信設備系統之可靠度，加速我國電信

等級設備之品質提昇，同時促使新技術與新應用服務的快速發展。

由於 ATCA 標準機架與 Carrier Grade Linux 皆可利用現有廠商提供的標準化產品，本計畫主要著重於 High Availability (HA) Middleware 與特定應用 EPON 的系統可靠度提升。以下對 HA Middleware 與 EPON 分別說明：

1. HA Middleware 標準發展與技術重點

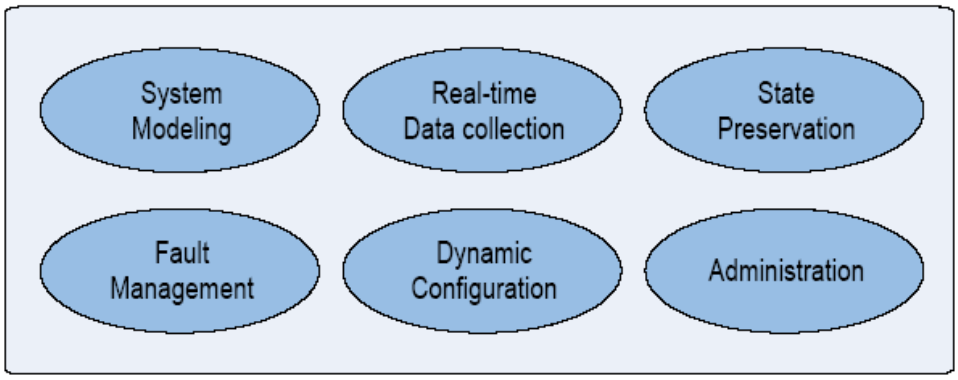
在開放架構的演進趨勢下，High Availability Middleware 的標準化已經逐步進行中。其中 Service Availability Forum (SAF) 組織就建構及制定了 Middleware 的開放標準架構與相關介面的標準。如圖一示，HPI(Hardware Platform Interface) Specification 介面規格定義了 Middleware 與下層硬體平台管理的介面標準，AIS(Application Interface Specification) 定義了 Middleware 對上層應用軟體的介面標準，而正在制定中 SMS(System Management Specification)則是定義了系統管理介面標準。



圖一、Service Availability™ 介面示意圖

SA Forum 所建構的 HA 模式，是採用 Clustering 及軟硬體 Redundancy 的方法，作為支援 HA 功能的技術基礎；而其所制定的 Middleware 標準，不但可符合 Cluster、Server、Database 等一般企業商用的高可靠度需求，更含括電信等級的 HA 網路、電信設備的需求特性與結構。其目的就是要推展標準化的 Middleware 開放架構，藉由模組化的架構設計與標準應用介面的提供，讓網路設備廠商與 HA 系統開發者能很容易且有效地於現有的(Commercial Off-the-Shelf, COTS)各種符合標準的產品組件(Building Block)中，選用合適的 HA Middleware、HA Hardware 等，加以整合開發各種 HA 應用產品與系統，以降低成本與縮短開發時程，並提供連續、不中斷的服務支援。本計畫在 EPON 平台上的研究成果，也可應用於其他類型設備，如 WiMAX、Metro Ethernet、Storage 等。

一般而言，支援 HA Middleware 的主要功能包含了：System Modeling、Real-time Data Collection、State Preservation / Data Checkpointing、Fault Management、Dynamic Configuration、Administration 等，如圖二所示。各個功能分述如下：



圖二、支援 High Availability 系統之 Middleware 功能示意圖

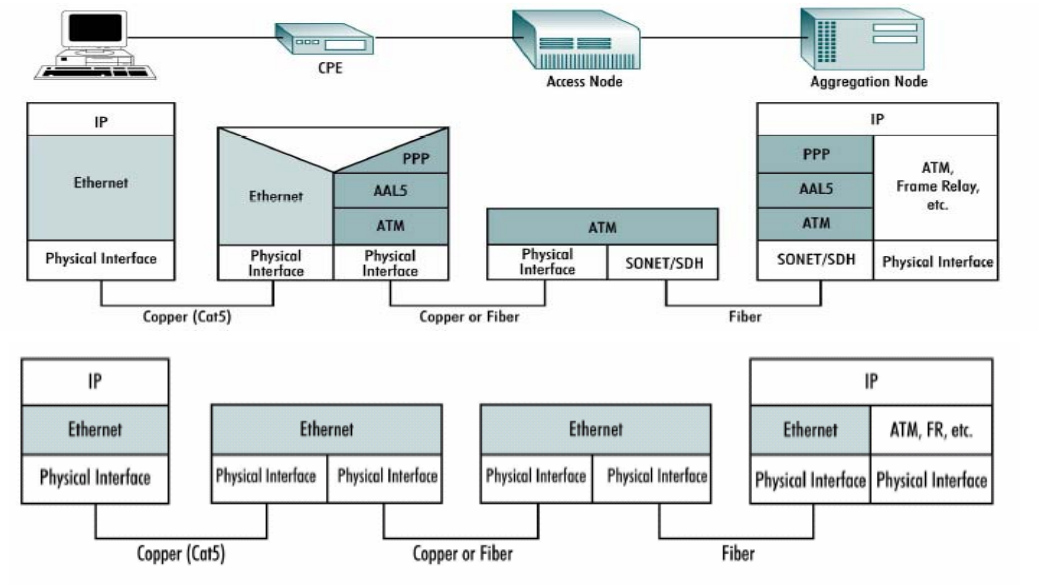
- 系統模型建立(System Modeling)主要是配置、設定(configuring)，並維護整個系統的狀態，用來呈現系統中所有被管理的軟硬體元件與資源(resource)，包括服務單元之建立，備援(redundancy)關係、群組關係，以及相互間的依存關係(dependency)之建立，active/standby 角色的訂定等，以便作為 middleware 可用度管理的依據。
- 即時資料收集(Real-time Data Collection)須持續即時的收集每個 resource 的狀態資訊，如環境狀態、運作效能等管理資料，以得以對 resource 進行後續的監控與管理。
- 狀態保存(State Preservation)與資料查核點建置(Data Checkpointing)，將不斷地複製異動資料及應用系統的狀態資料，儲存至備用端(standby)，進行運作資料及狀態資料的同步(Synchronization)，以便在 active 端發生故障時能即刻迅速地切換至 standby 端，由 standby 端來接手應用系統的運作。
- 錯誤管理(Fault Management)是對不預期的錯誤，進行偵測、診斷、隔離，以及復原、修護的處理措施(Detection，Diagnosis，Isolation，Recovery，and Repair)。
- 配置組態的動態管理(Dynamic Configuration)主要是動態的調整系統元件及 resource 的 configuration 與 dependency，透過 heartbeat 機制，隨時查驗 cluster 成員(membership)及元件的存活狀態(health)，並做相關的處理與角色的調整。
- 除了上述這些自動管理與自我管理的功能以外，Middleware 同時提供使用者管理功能(Administration)，管理存取與管控系統的權限。

2. EPON 標準發展與技術重點

EPON 是一種基於被動式光纖網路架構（Passive Optical Network，PON）的乙太網路傳輸技術，它採用在一根光纖上以波分多工(Wavelength Division Multiplex，WDM）技術實現點對多點雙向通信，將乙太傳輸協定引進接取網路，完成一個從端點到端點，單一傳輸技術的寬頻網路環境（見圖三），而免於處理各種不同協定間之轉換，同時順應了下世代網路（Next Generation Network，NGN）IP 化之發展趨勢。

傳統乙太傳輸方式接取網路需多個協定轉換

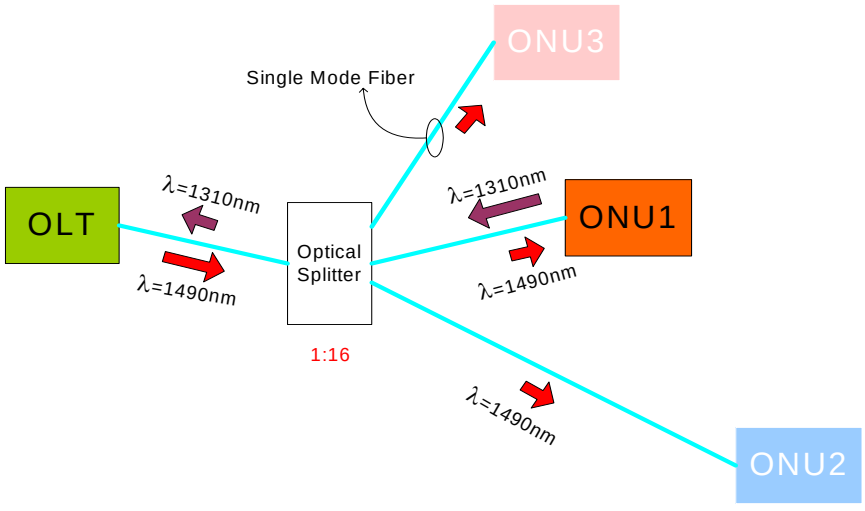
從端點到端點，均採單一乙太傳輸方式



Source ： Light Reading

圖三、用戶端-局端之傳輸協定示意圖

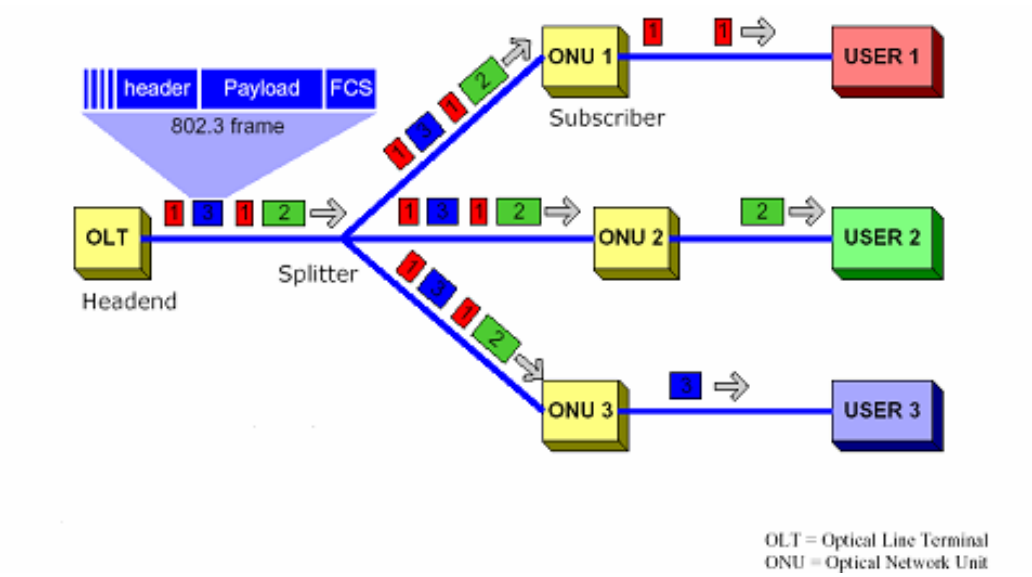
EPON 系統基本上由局端設備（Optical Line Terminal，OLT）、遠端設備（Optical Network Unit，ONU）和被動式的光分歧器（Passive Optical Splitter）組成（見圖四）。在傳輸速率及傳輸距離上，EPON 可以支援 1.25Gbps 雙向對稱速率，最大傳輸距離可達 20 公里，分歧器之分歧數量以 1：16 居多，但目前技術上則已達 1：32。至於在光波長的運用上，EPON 使用了 1310nm、1490nm 及 1550nm 三個波長，其中 1310nm 波長係負責承載由 ONU 端往 OLT 端傳送的上傳數據資料，而 1490nm 波長則係負責承載由 OLT 端往 ONU 端傳送的下傳數據資料，另外 1550nm 波長則負責承載 CATV 的類比電視頻道。



圖四、基本 EPON 網路架構

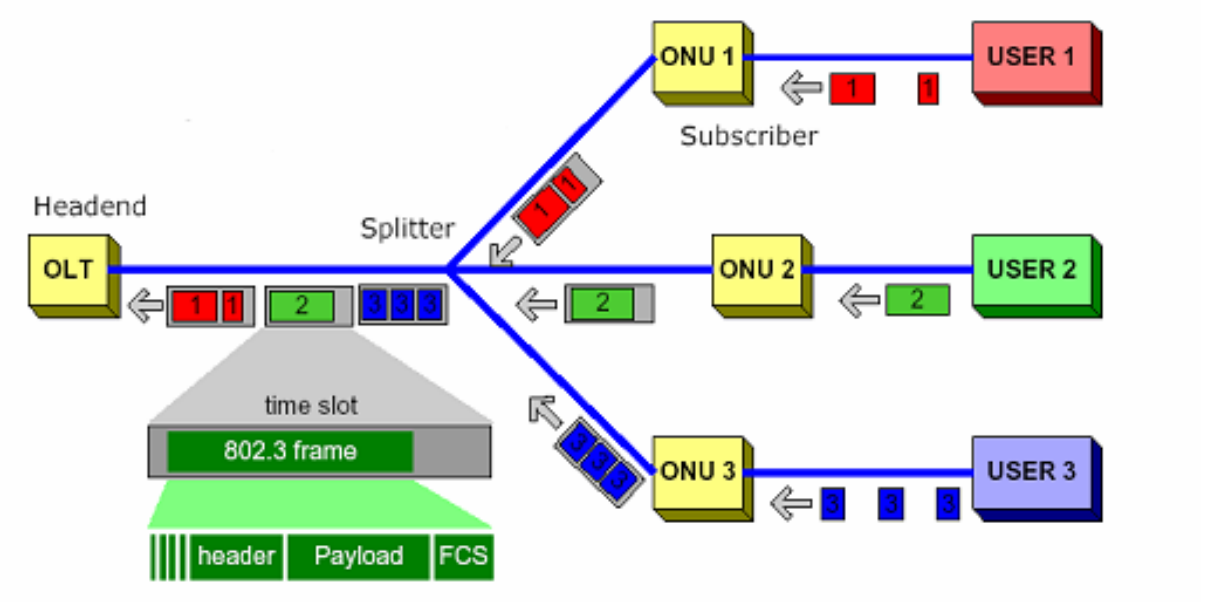
在 OLT 端，利用 WDM 將下傳（由 OLT 端往 ONU 端傳送）方向的 1490nm 及 1550nm 這 2 個波長耦合到單模光纖（Single Mode Fiber）上，同時也將上傳（由 ONU 端往 OLT 端傳送）方向的 1310nm 這個波長耦合給 OLT 的光模組（Optical Transceiver）接收。由於 OLT 至 ONU 下行方向是以點對多點（P2MP）的廣播（Broadcast）方式，並交由

用戶接收端擷取所需的訊號（見圖五），因此相當容易提供 video multicast / broadcast 的服務;OLT 至局端的上行方向,則利用 Gigabit Ethernet 介面連接。OLT 同時也是 EPON 的網管的主要控制點，除了提供 OLT 本身的網管功能外，也可進一步代管所連接的多個 ONU。



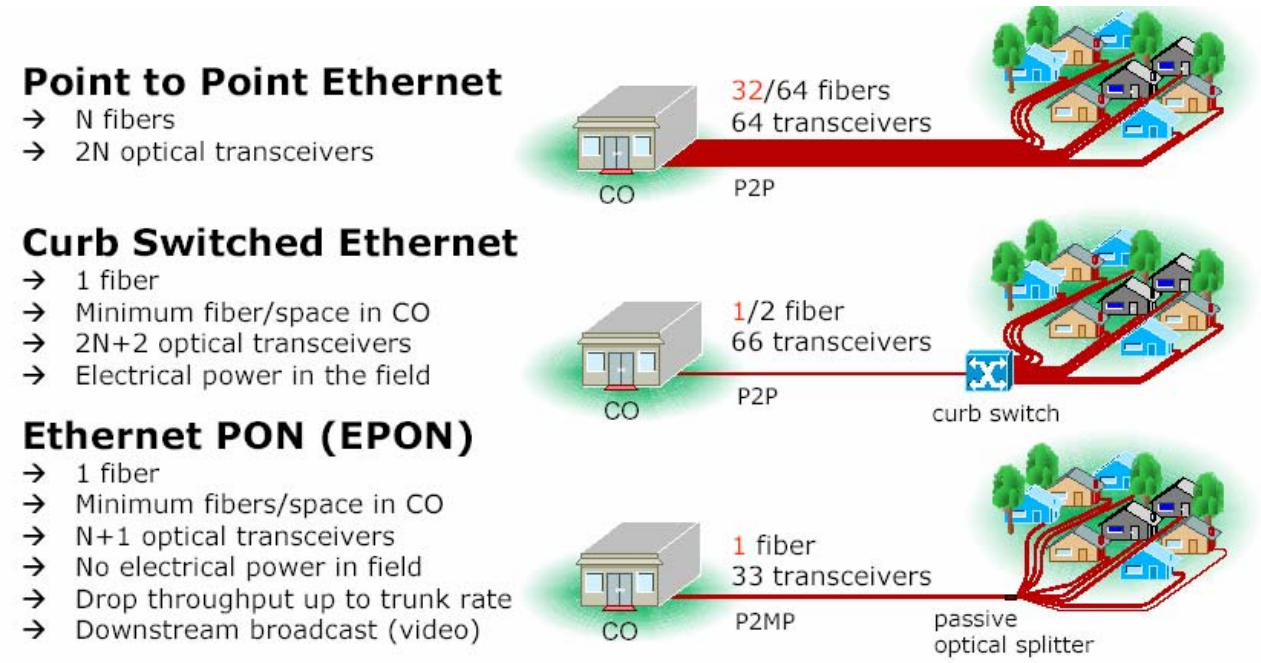
圖五、EPON 下行機制

在 ONU 端，WDM 負責在下傳（由 OLT 端往 ONU 端傳送）方向將 1490nm 及 1550nm 這 2 個波長分別耦合給 ONU 的光模組（Optical Transceiver）及 CATV Receiver 模組，同時也負責在上傳（由 ONU 端往 OLT 端傳送）方向將 1310nm 這個波長耦合到單模光纖上。至於 ONU 上行傳輸的處理方式則十分複雜。ONU 上行傳輸是以點對點（P2P）方式，按照 OLT 中的控制機制進行，採用分時多工（Time Division Multiplex，TDM）方式，搭配動態頻寬配置（Dynamic Bandwidth Allocation，DBA）機制，分配每個 ONU 專用的傳輸時槽（Time Slot），可以防止來自不同 ONU 的數據傳輸產生碰撞（見圖六），並提供服務品質（Quality of Service，QoS）保證。



圖六、EPON 上行機制

由於 EPON 能提供 FTTH、FTTB 或 FTTC 的架構，以最經濟的光纜數與光收發器數目來達成服務相同的客戶數（見圖七），並配合 DBA 來滿足服務品質需求，因此可作為提供語音、數據及影像的應用及服務之全方位業務平台。日本 SoftBankBB 在 2004 年 7 月進行了 30 萬門的 EPON 設備採購標案。日本 NTT 則宣佈自 2005 年起將停止其它 FTTH 佈建方案(如 MC、BPON 等)，開始大規模 EPON 網路建置，以提供 FTTH 服務，預計至 2010 年將投入超過 400 億美元。而在大陸方面，在 2004 年 7 月的武漢市第一期 FTTH 示範工程中，EPON 擊敗其它技術，獲選並成功建置服務 158 戶用戶，現正進行第二期 2000 戶的 FTTH 示範工程建置。2005 年底武漢市將進行第三期 50000 戶的 FTTH 示範工程建置，提供資訊、音頻與視頻等三網合一的服務。在台灣的部份，中華電信 2003 年起進行一連串之 EPON 設備功能測試、光纖到家從用戶迴路基礎網路建置、服務供裝、維運與障礙查測技術評估，預計 2005 年將進入小量商用階段。



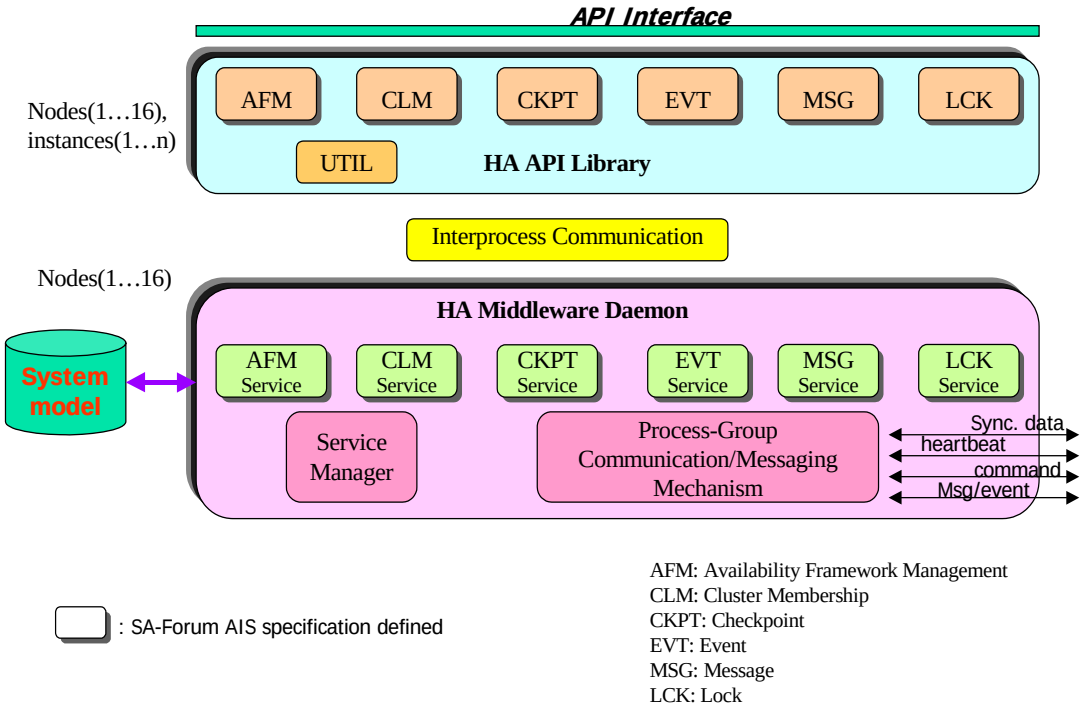
圖七、EPON 為 FTTx 之較經濟的解決方案

第2章、 研究目的

有鑒於 EPON 及其他電信等級寬頻網路設備的廣大市場潛力，以及高可靠度技術對於國內通訊產業提升的關鍵性，本計畫將以 EPON 系統為平台，針對 High Availability Middleware 進行研究開發與效能分析，並建置 Carrier-Grade High Availability EPON OLT 系統的測試驗證平台。兩工作項目分述如下：

2.1 支援電信等級通訊設備之 High Availability Middleware 研究開發與效能分析

SA Forum 除了針對 HA Middleware 的各個介面訂定標準外，對於 Middleware 本身也做了基本的描述。由於與上層應用服務介接的是 AIS 介面，因此由 AIS 介面往下的內部功能與架構描述較完整，如圖八所示。其中如：Cluster Membership、Checkpoint、Event、Message、Lock 等功能都是用於多個 CPU 間從屬關係的管理與資料的交換與同步，對於設備能否在元件故障時迅速的偵測與切換，且維持一致的管理與運作資訊，具有決定性的影響。各種機制在設計時需考慮到效能(故障切換及復原時間)、成本(CPU 運算與網路資訊交換)、彈性(支援 N+1、N+M 等各種保護模式)。在 EPON 及其他寬頻電信等級設備的特定平台下，由於處理器運算能力、記憶體空間等 embedded 系統資源的侷限性，以及上層網路協定間彼此的關聯性，如何就資源與效能作最好的權衡



(tradeoff)，甚至開發更好的機制與架構，值得深入分析研究。

圖八、HA Middleware 軟體架構設計

本工作項目之電信等級 HA Middleware 研究開發，先將依循 SA Forum 的標準，開發符合 SA Forum 規格的 HA Middleware。主要將提供標準 AIS 應用介面，基於符合 POSIX 的 Linux 等作業系統平台，以及 ATCA 硬體平台，開發嵌入式的 HA Middleware，研發各項核心及關鍵技術，預定系統的初步效能將小於 200ms 的故障切換及復原時

間，並能支援 N+1、N+M 等一對多的 redundancy 保護模式。

由於符合電信等級的要求，必須考量網路、電信、通訊設備與系統的需求，並配合 embedded system 之特性，因此在技術上計有下列重點需要考慮並予以突破及解決：

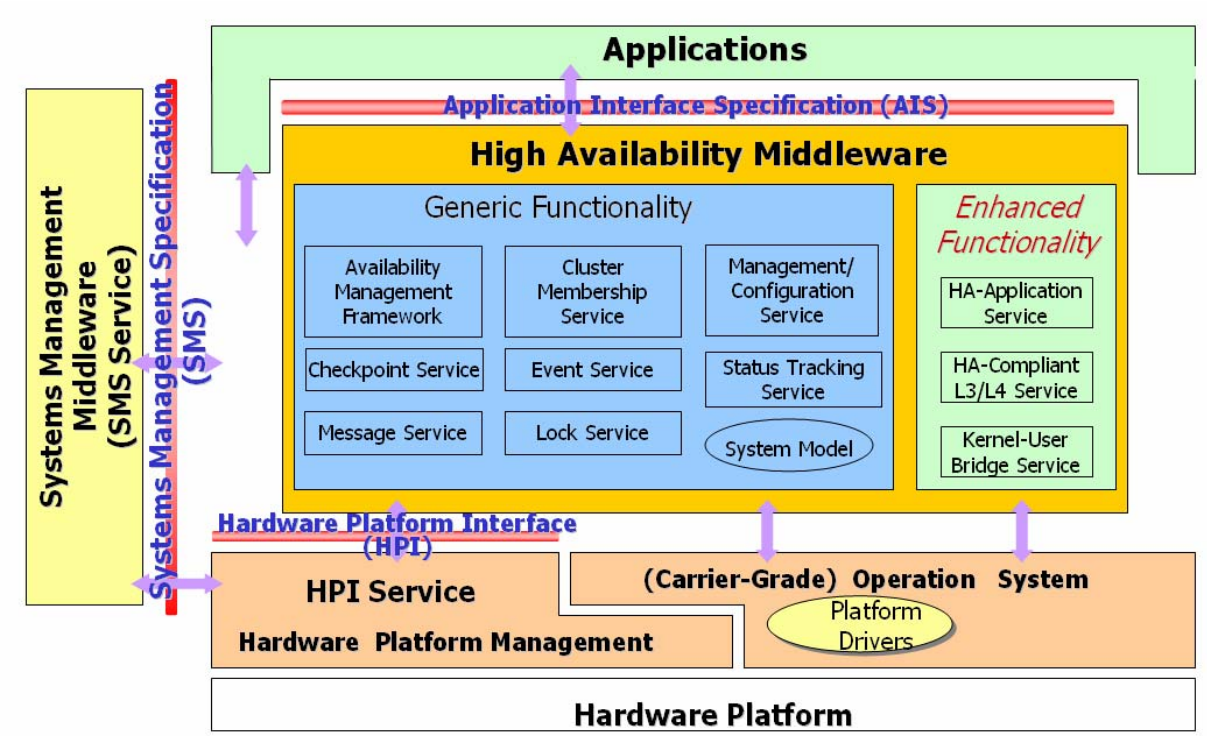
1. 效能及速度需求：為達到電信等級的要求，需要有高效率及快速的系統效能，並需兼顧到所造成系統的額外負擔(overhead)不宜過多，主要需求項目如下：
   - 快速的故障偵測(failure detection)及反應處理
   - 快速、且自動的故障切換(failover)機制，且能支援狀態資料之同步移轉(stateful)，以提供連續服務的可用度能力。
   - 快速、高效能的訊息傳輸通訊機制

2. 有限的系統資源限制：受到包括處理器運算能力、記憶體空間等 embedded 系統資源的侷限性，middleware 對系統資源的利用需朝最佳化設計。

3. 記憶體資料同步(In-memory data synchronization)的需求：為支援完整的 stateful failover，in-memory 的資料同步需滿足即時性與大量資料的需求，並需達成其完整性、正確性與一致性，因此，設計一個可靠、strong atomicity 的 checkpoint 機制是極其重要的。

4. 網路協定(Protocol)/應用服務功能的 HA 需求：包括連續的服務執行、協定的 dependency 與復原方法(recovering algorithm)等。

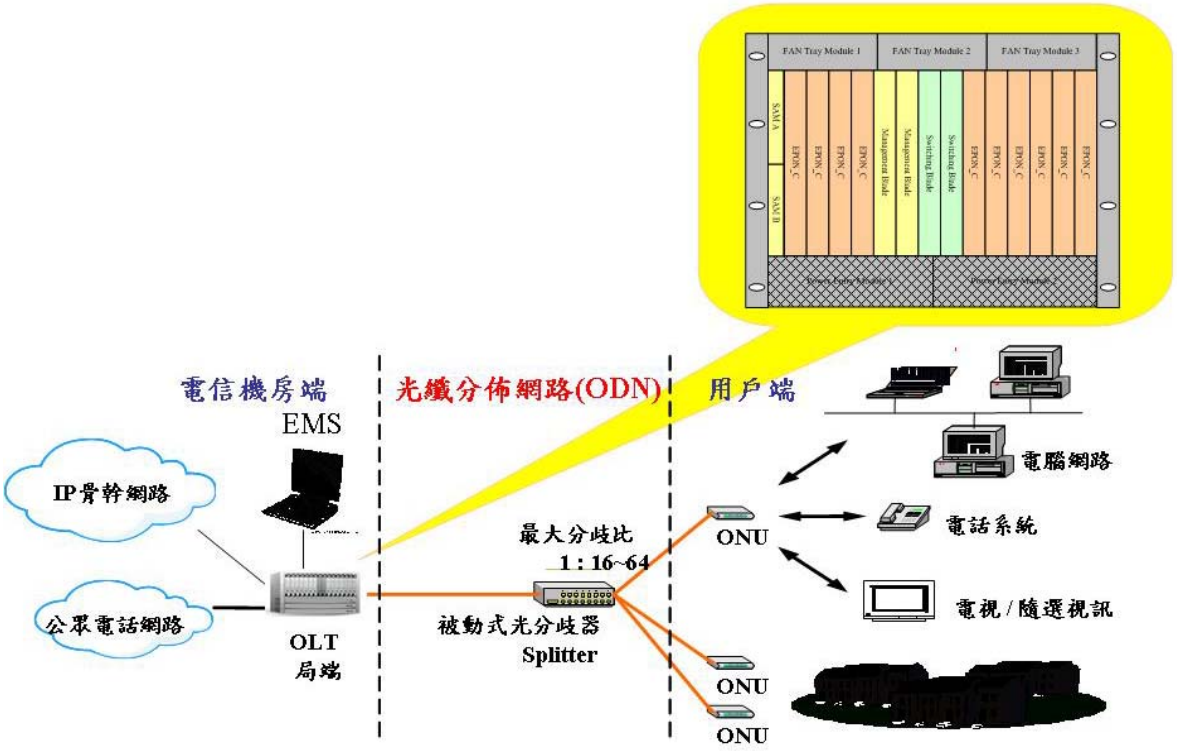
因此，本工作項目之電信等級 HA Middleware 的研究開發，不僅需完成符合 SAForum 原本規範的 Generic Functionality，更重要是研究開發如圖九所示現仍未有的關鍵 Enhanced Functionality 以滿足上述四點考量，並將相關研發成果貢獻至 SAForum 以使 HA Middleware 的規範更加完整。本年度計畫分兩大研究主題來達成此一目標，各主題的研究目的、文獻探討、研究方法、結果與討論等內容分別詳細說明於第 3、4 章。



圖九、HA Middleware 的 Generic 及 Enhanced Functionality

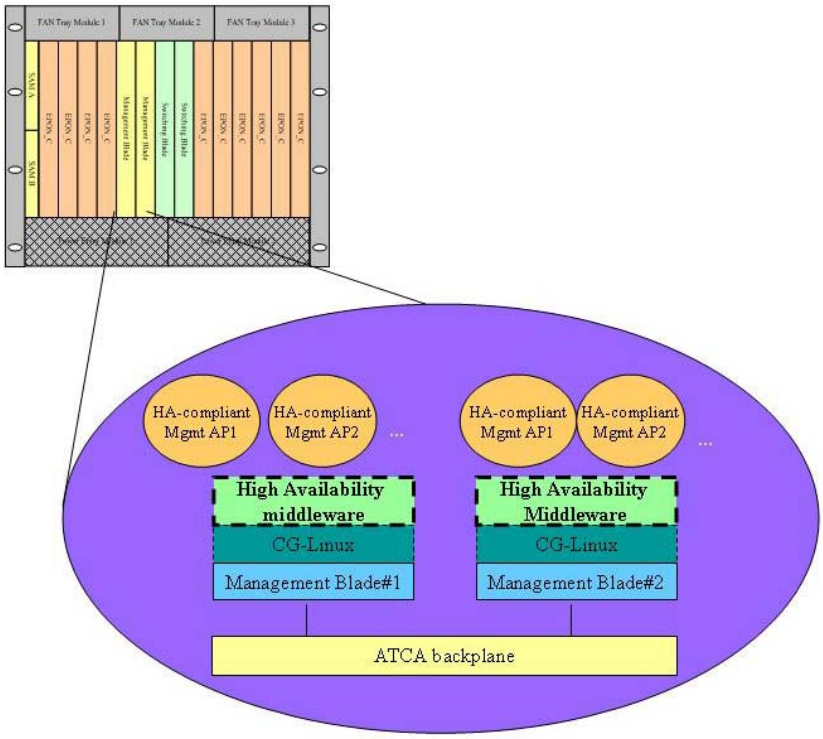
2.2 符合 Carrier-Grade High Availability EPON OLT/Server 系統之建置與測試驗證

本計畫所選擇的開放架構 EPON 系統常見的組態如圖十所示。主要組成元件包括了三種不同的卡板：(1)EPON Blade：是符合 IEEE 802.3ah 的 EPON OLT 卡板，每個 EPON OLT port 可以連接到 16 個用戶端 ONU；(2)Management Blade：提供給系統管理員來管理 EPON OLT blade 與 Switch Blade；(3) Switch Blade：提供整個系統的 data path switching center，並且提供上行的連接介面。



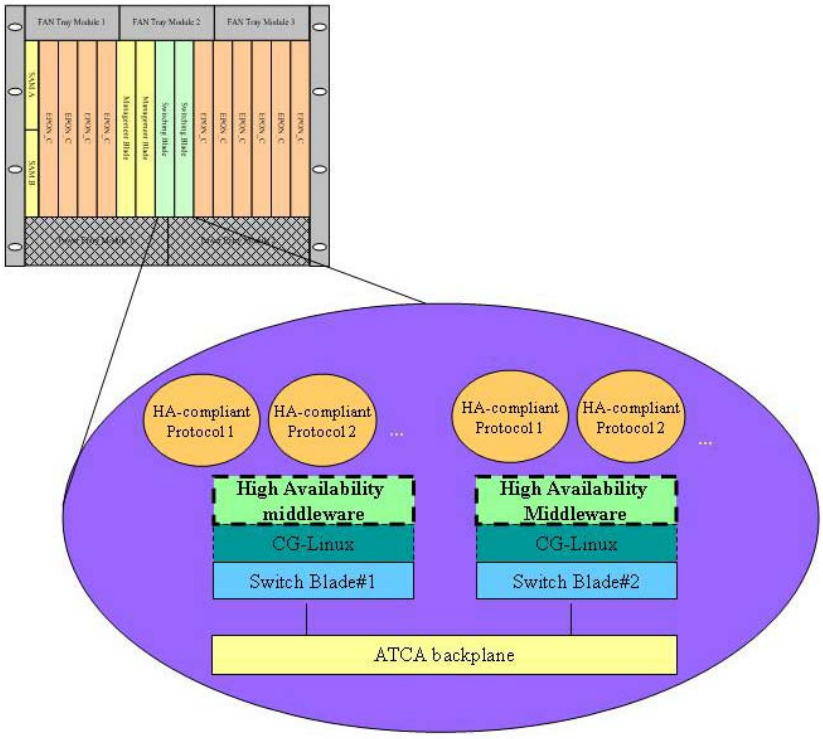
圖十、ATCA-based EPON 局端系統與服務網路的建置

在整個 EPON 局端系統中，management blade 及 switch blade 必須提供備援保護。若只有單一的 Management Blade 且發生故障，整個系統就無法管理；若只有單一的 Switch Blade 且發生故障，整個系統就無法送收資料；這都是局端設備所不允許的。利用 ATCA 架構提供的備援硬體機制，再搭配 Middleware，將可達成備援硬體間的協調與資料同步。在 Management Blade protection 方面，其作法如圖十一所示。兩張 Management Blade 將透過 ATCA 背板來做 1:1 的保護機制。當一張 Management Blade fail 時，能切換到另一張 management blade 繼續執行。在每張 management blade 上，將執行 carrier grade Linux 之作業系統，以提供一個 reliable OS 的環境。在此 carrier grade Linux 上，將經由 Middleware 來提供 management protection 的功能。此處我們將以 SNMP agent 作為管理功能方面分析研究與驗證的案例。



圖十一、EPON System Management Blade Protection

另一方面，在 Switch Blade protection 的作法如圖十二所示。與 management blade 作法相似，兩張 S witch Blade 將透過 ATCA 背板來做 1:1 的保護機制。當一張卡 fail 時，能切換到另一張 switch blade 繼續執行。在每張 switch blade 上，將執行 carrier grade Linux 之作業系統，以提供一個 reliable OS 的環境。在此 carrier grade Linux 上，我們將以 IEEE 802.1d Spanning Tree 作為資料交換功能方面分析研究與驗證的案例。



圖十二、EPON System Switch Blade Protection

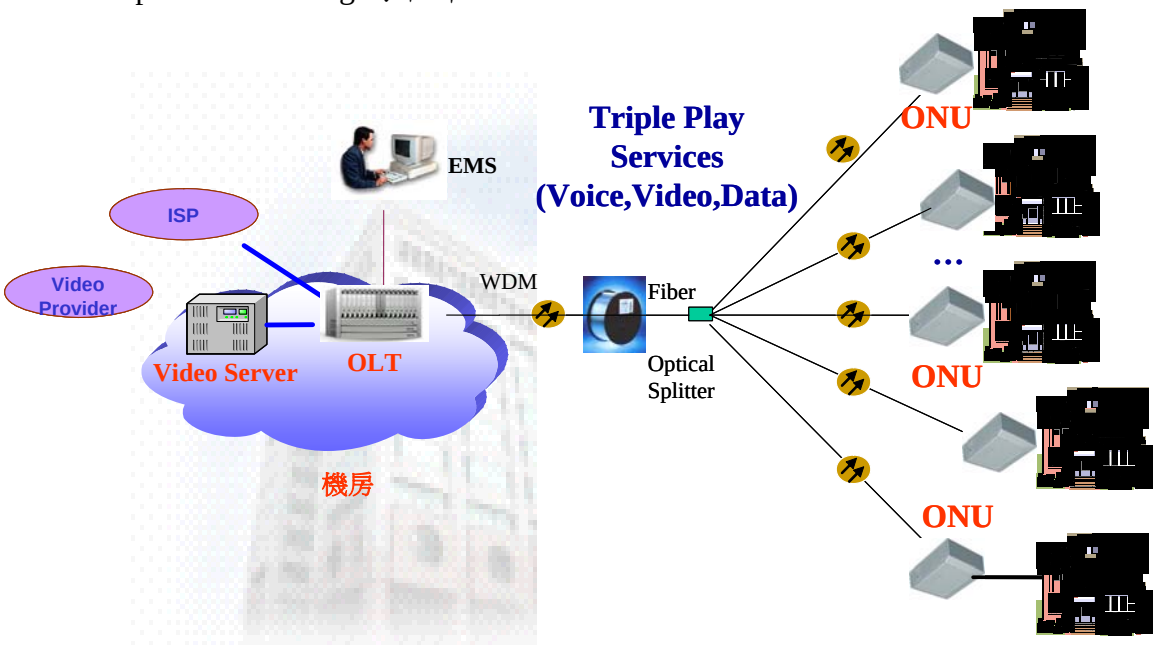
為驗證架構設計與效能分析的結果，本計畫將採購符合開放平台標準的 EPON 設備，來建置一個試運轉與可靠度測試的環境，如圖十三所示。在此試運轉環境中，將提供 Voice, Video 及 Data 等 triple play services，以測試 EPON OLT 及 ONU 是否能夠很穩定的提供這些服務。在可靠度方面的測試重點包括了：

- Management Blade Protection /failover：

  當一張 management blade fail 時，能順利切換到另一張 management blade 且不會造成 internal 和 external control data exchange 的中斷。

- Switch Blade Protection/failover

  當一張 switch blade fail 時，能順利切換到另一張 switch blade 且不會造成 data plane forwarding 的中斷。



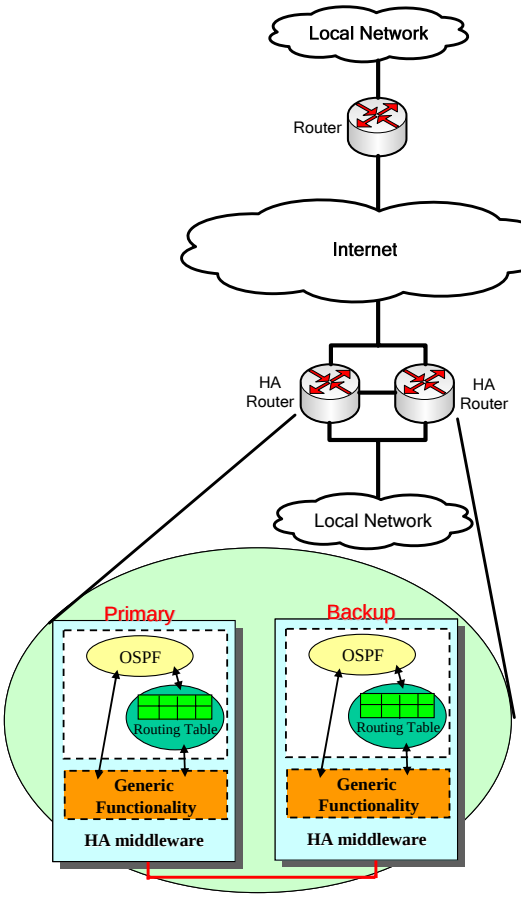
圖十三、High Availability EPON OLT 系統 Field Trial 環境

第 3 章、Fast recovery mechanisms of routing paths for HA middleware

3.1 研究背景及文獻探討

由於近年來寬頻網路的蓬勃發展，促使網際網路服務商對於基礎網路的建設更加重視，此外網路的使用也從最基本的資料交換到即時的影音傳送，如視訊會議(video conference)、網路電話(Voice IP)，網路通訊正朝向多媒體的資訊網路發展，隨著人們對於網路傳輸的日益依賴，基礎網路上之路由器的可靠度及連續性服務的需求也越來越迫切。現階段當網路路由器若發生服務中斷，處理方式可能由網管軟體透過 SNMP (Simple Network Management Protocol)協定偵測，再通知給網管人員，再由網管人員解決問題，如此，網路服務的中斷時間可能達數十分鐘甚至數小時，此處理方式對於電信等級之網路服務品質是不夠的。因此，本研究將根據 SA Forum 所提出之 HA Middleware Generic Functionality 並加入 Enhanced Functionality 相關服務讓基礎網路上之網路路由器能提供更高的網路可靠度。

基礎網路是由一個以上之路由器所組成的，每個資料封包將由路由器轉送至正確的網路區段，若路由器服務中斷，則將造成資料封包將無法被正確的轉送至正確的網路區段；但若在基礎網路上的路由節點有備援之路由器，如圖十四中 HA Router，在一般時候只會有一個路由器服務，負責轉送封包，當主要路由器服務中斷時，另一個備援之路由器將會立即接手封包轉送之工作，因此，資料封包依然可以正確的被轉送至正確的網路區段。



圖十四、高可靠度之路由系統示意圖

因此，在圖十四中 Backup Router 如何立即接手封包轉送之工作是能否達到電信等級高

可用度要求之重要部份。本研究擬結合本年度所實現的 Novel K-U bridge schemes for HA middleware 為基本架構，針對 Kernel space 中的 routing table 提出一個共同介面，讓各個不同的 routing protocol 可以透過此介面來進行 routing table 的管理以及備援，以達到高可靠度之目的來進行研究討論。並希望其效能能超越其他不同作法的國內外相關研究，簡要敘述如下：

- Graceful Restart, RFC 3623

若網路中的 Router 暫時中斷服務時，Neighbor Router 將不會立即更新 Routing Information 以及發送新的 Routing table 給網路上所有的 Router，而是會等待 LSRefreshTime(1800 秒)之後，若原本服務的 Router 依舊無法服務，才會認定其已經中斷服務，再重新計算路由資訊並且向網路上所有節點發送新的路由資訊。若在 LSRefreshTime 之間，中斷服務的 Router 可以恢復服務，則 Neighbor Router 依舊將其視為正確的路由節點，因此網路上所有路由節點的 Routing Table 皆可全部保留無需重新計算。

缺點：

- ✓ 網路中所有的 Router 皆必須支援 Graceful Restart 協定。
- ✓ 在 Graceful Restart 進行期間，網路拓蹼不能有所改變，否則 Graceful Restart 就算失敗，網路上所有 Router 必須重新計算路由資訊。
- ✓ 當 Neighbor Router 沒有接收到 Router 的 HELLO 訊息才能得知路由節點的服務中斷，HELLO 訊息的預設發送間隔為 10 秒鐘，因此當路由節點中斷服務時，Neighbor Router 將隔一段時間才能進行 Graceful Restart。
- ✓ 因為有 Router 中斷服務，因此網路上有些資料封包無法被轉送至正確的目的地。

- VRRP (Virtual Router Redundancy Protocol), RFC 3768

利用多個 Router 構成一個路由器組，由此路由器組來轉送網路上的封包，在任一時間內，此路由器組中只會有一台 Router 用來服務轉送封包，路由器組會共同擁有一組虛擬 MAC 以及 IP 位置，因此，在此路由器組所服務的範圍之內的網路節點都會傳送封包至此虛擬 MAC 及 IP，當目前服務的 Router 中斷服務時，路由器組會決定出一 Router 用來接手先前 Router 的工作。

缺點：

- ✓ 本協定為硬體備援，所以需要有特殊的硬體來支援 VRRP。
- ✓ 因為在同一時間內，只有一個路由節點服務，因此，不具有負載平衡的功能。

- HSRP (Hot Standby Routing Protocol), RFC 2281

同 VRRP，但此規格是 Cisco 公司的專利，若要使用須先取得同意。

- Redundant Node

在網路的路由節點上，利用 Redundant Router 作備援。當原本服務的 Router 中斷服務時，Redundant Router 就會立刻被啟用，用來接手原本路由節點的工作。

缺點：

- ✓ 需要多餘的設備作備援
- ✓ Failover 之後，所有的路由資訊必須重新計算。

✓ 因為路由節點組並沒有交換任何 Routing 資訊，因此無法實行負載平衡。

✓ Failover 之後，整個網路 Routing Table 重新計算所花之成本過高。

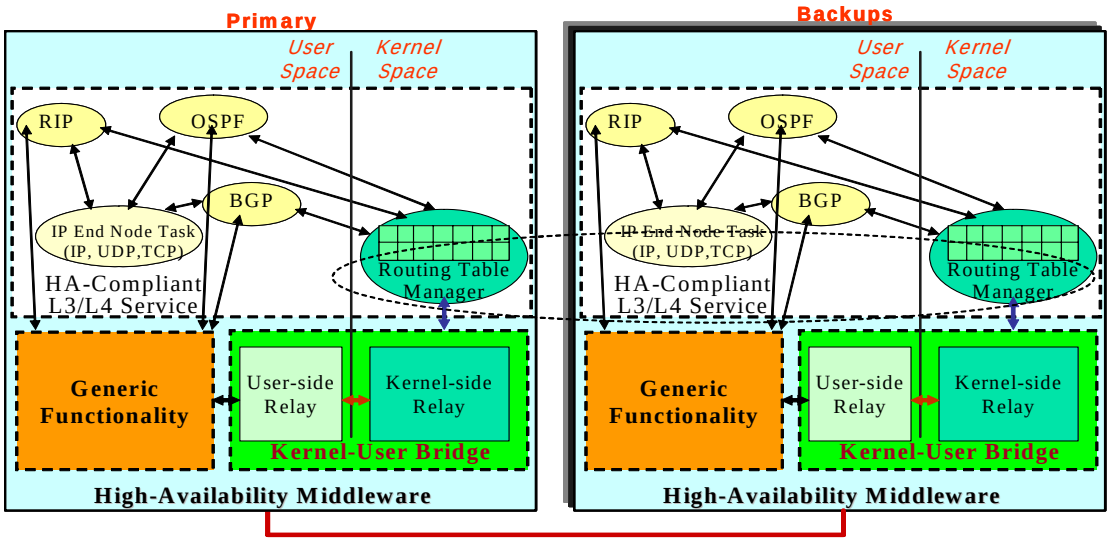
3.2 研究目標

當基礎網路達到穩定狀態時，每一個 Router 都將有一份穩定狀態的 Routing Table，Router 根據此 Routing Table 可以將封包轉送至正確的網路區段，為了提升網路可靠性及無中斷的網路服務能力，本部分擬將在 HA Router 中設計一介面，讓 user space 的程式，如 routing protocol 可以透過此介面存取存在於 kernel space 中的 routing table，以便對 routing table 進行備援以達到網路的無中斷服務。

雖然各個不同 routing protocol 的演算法不同，但是其目的皆為計算出一個最佳路徑的 routing table，因此，若能夠設計開發出一個介面，則每一個 routing protocol 皆可使用此介面對 routing table 作備援存取，因此本研究所期待之無中斷網路服務，將可不受限於某些特定之 routing protocol，如此，將可提高系統的可適用性。

如圖十五所示，本研究將擬制定出一共同的介面，以便供 user space 程式呼叫使用，如圖中的 Generic Functionality；所設計出一 Kernel-User bridge 介面，用來接收來自 Generic Functionality 的指令，以便透過此 bridge 來存取位於 kernel space 的 routing table 資訊。

當能夠存取 Kernel space 之 routing table 之後，active 角色的 Router 便可利用 HA Middleware Generic Functionality 所提供之 checkpoint service 將 routing table 資訊更新備援至 standby。Active 與 standby 利用 HA Middleware Generic Functionality 所提供之 AMF service 便可偵測 active 是否仍可提供服務，以便讓 stanby 能夠 takeover 先前 active 所執行之轉送封包動作。



圖十五、Routing Table Manager redundancy 的架構設計

3.3 研究內容及方法

本研究擬結合本年度所實現的 Novel K-U bridge schemes for HA middleware 為基本架構，針對 Kernel space 中的 routing table 提出一個共同介面，讓各個不同的 routing protocol 可以透過此介面來進行 routing table 的管理以及備援，本年度將先選擇 OSPF routing protocol 設計，以達到高可靠度之目的來進行研究討論。以下分別說明 Novel K-U bridge schemes for

HA middleware 及 HA OSPF Routing 的架構設計：

3.3.1 Novel K-U bridge schemes for HA middleware 的架構設計與效能分析

本年度針對 System Model、Group Communication、Messaging mechanism 等核心技術進行更深入的探討，並提出 Novel K-U (Kernal-User) bridge schemes for HA middleware 的架構設計以實現在圖九中 HA Middleware Enhanced Functionality 的 Kernal-User Bridge Service。相關設計與效能分析結果分別說明如下：

A. System Model 之設計

Middleware 的管理與運作，必須依據整個系統的配置與組態。System Model 在設計上，必須包含以下各部分：

1. 系統配置與組態的描述：包括了所有被管理的軟硬體元件與資源(resource)之描述，含名稱/ID、編號、屬性等；服務元件、服務單元的群組關係；備援(redundancy)保護關係，以及 active/standby 的決定與委派方式；各元件及服務間的依存關係(dependency)等。

2. 組態的動態調整的支援：即時得知系統配置組態的動態變化，如軟硬元件的新增與消減等。尤其是須考量及支援硬體元件或卡板的 hot swap 熱插拔能力。

3. Active/Standby 角色管理：依所設定之 active/standby 的決定與委派方式，以及系統配置組態的動態變化，進行 Active/Standby 角色之調整與管理。

4. 服務元件與服務單元的群組關係及依存關係之管理：可將數個服務元件群組成一個服務單元，視為單一個服務單位來進行管控，若某服務單元中的一服務元件停止運作，則同屬該服務單元的其他服務元件也將自動被停止。另外，也要支援各服務間的 dependency 關係，例如在 MPLS 的應用中， 流量工程(traffic engineering) 功能需使用路由(routing)功能所產出的路徑表，因此當 active node 中的路由功能發生故障時，該 node 的 MPLS 流量工程功能也無法正常運作，因此應該與路由功能一起切換至另一個 standby node 來執行。

B. Group Communication、Messaging mechanism 之架構設計與效能分析

由於 HA Middleware 的各服務模組，必須依賴 IPC(Inter-Process Communication) 機制來進行相關的資料傳輸，若能有一支援 process-group multicast 的 IPC 機制，將能使整個 middleware 的設計更加的簡易、有效。因此，設計及建置一個分散式計算架構(distributed computing framework)，提供一有效率的群組通訊及訊息傳輸(group communication & group messaging) 機制，將是設計整個 HA Middleware 的重要關鍵。這個機制的主要項目如下：

1. 傳輸(delivery)服務機制：將設計一 group communication 協定，以提供可靠的多點傳送(reliable multicast)能力，使其具順序性、整體性(integrity & atomicity)與一致性，並達成高效率且正確的大量資料傳輸性能，以支援 process-group 及 node-group 的傳輸需求。利用本機制，將可直接支援 HA Middleware 的 event service 及 message service 等服務，也支援 checkpoint service 進行對 cluster nodes 的資料同步傳輸(data synchronization)，並對 availability management 等 middleware 服務模組進行相關資料傳輸，提供一個高階的通訊機制。

2. Membership & Configuration 管理：為進行 group 群組通訊，通常需要掌握對 group 成員及配置(membership & configuration)的最新資訊與管理，以隨時得知最新的群組成員加入或離開等狀態。透過此功能，將可直接支援 HA Middleware 的 cluster membership service 及 availability management 等服務。

C. Linux-based Kernel/User Spaces 橋接(bridging)的架構設計

開放架構採用的 Linux 作業系統內部區分為 user space 與 kernel space，前者主要為一般的應用軟體模式，其開發除錯較易，工具與環境之支援也較充足，後者主要為驅動軟體、系統程式的運用模式，執行效率較佳，但於開發除錯上也較困難。目前 HA Middleware Genric Functionality 的實現上普遍都在 user space 進行，但不少寬頻設備基於效能與硬體控管的考量，常會將功能實現在 kernel space。為了讓 kernel space 各模組能有效的利用 HA middleware Generic Functionality 來滿足系統可靠度的要求，如圖十六的一個有效率的 Kernel–User Bridge 是不可或缺的。

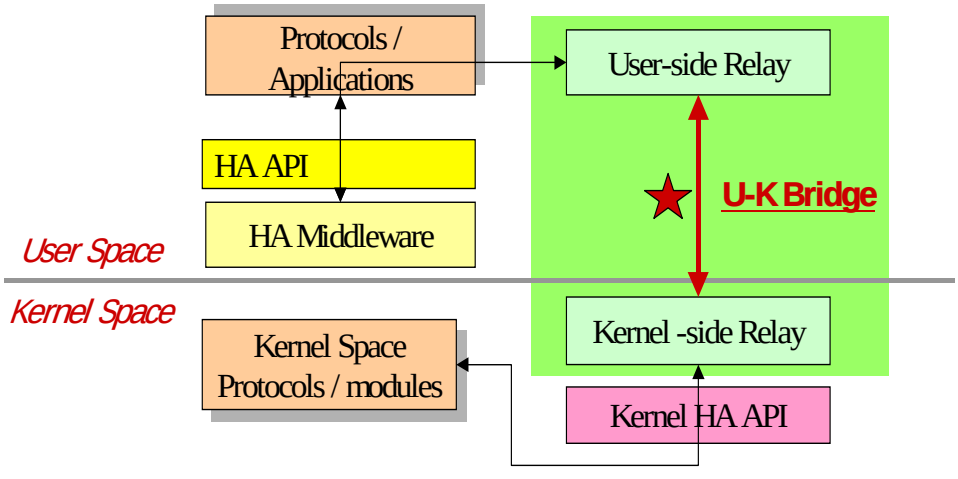
本年度依據上述三項架構設計要點，所設計出的 Novel K-U bridge schemes for HA middleware 之運作架構如圖十七所示，主要包含兩大部分：

- User-side Relay：包含一 "User Relay Manager" 程式。
- Kernel-side Relay：包含了"Kernel Relay Manager" 及 "Kernel Interface Module" 等二項模組 module。

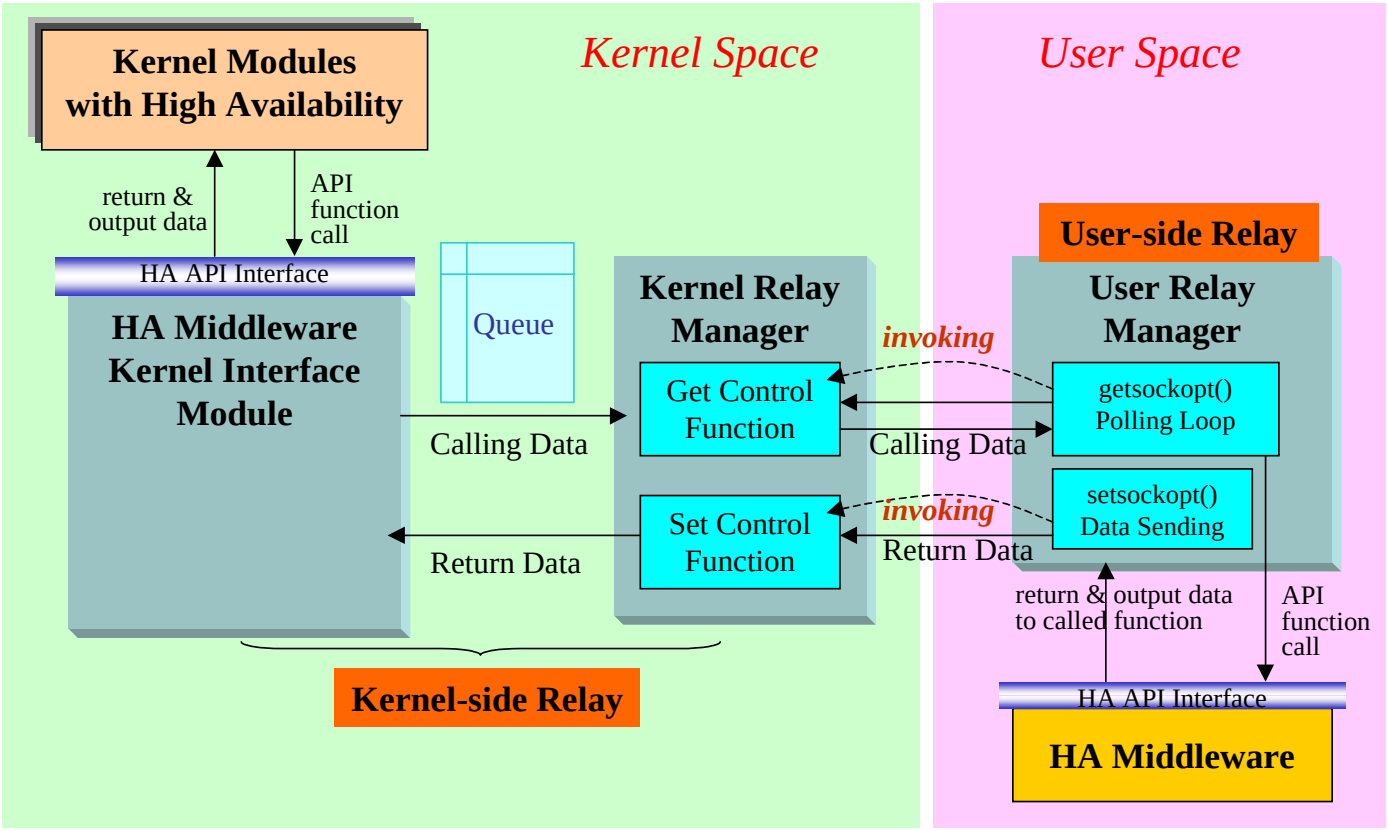
其中，"user relay manager" 是以 polling 的方法去向"kernel relay manager" 詢問以取得所傳送之資料，其計有 3 種可行的設計方式：

- Continue Polling
- Pause xx msuc per polling
- Pause 10 msec per xxK polling times

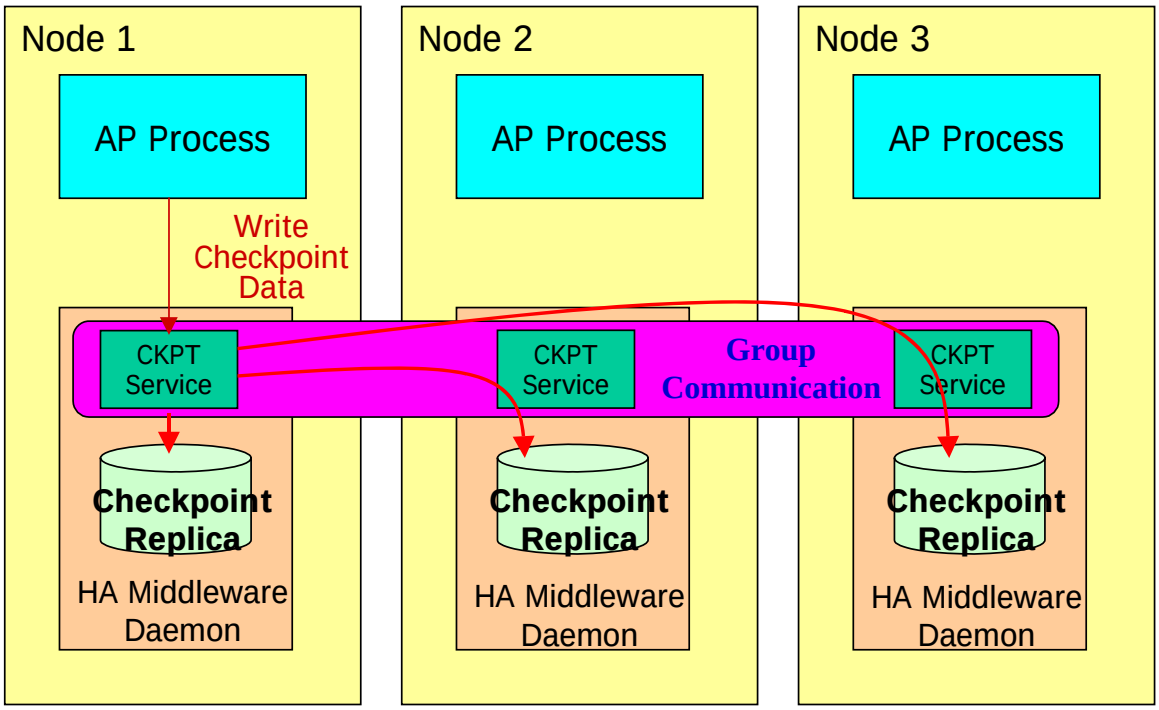
本年度已實作完成圖十七所示的 Novel K-U bridge schemes for HA middleware 設計，並採用 HA Middleware Generic Fuctionality 中的 "Checkpoint" service API function 服務功能來架構系統測試程式 Benchmark，其運作如圖十八所示。



圖十六、Kernel–User Bridge 機制示意圖

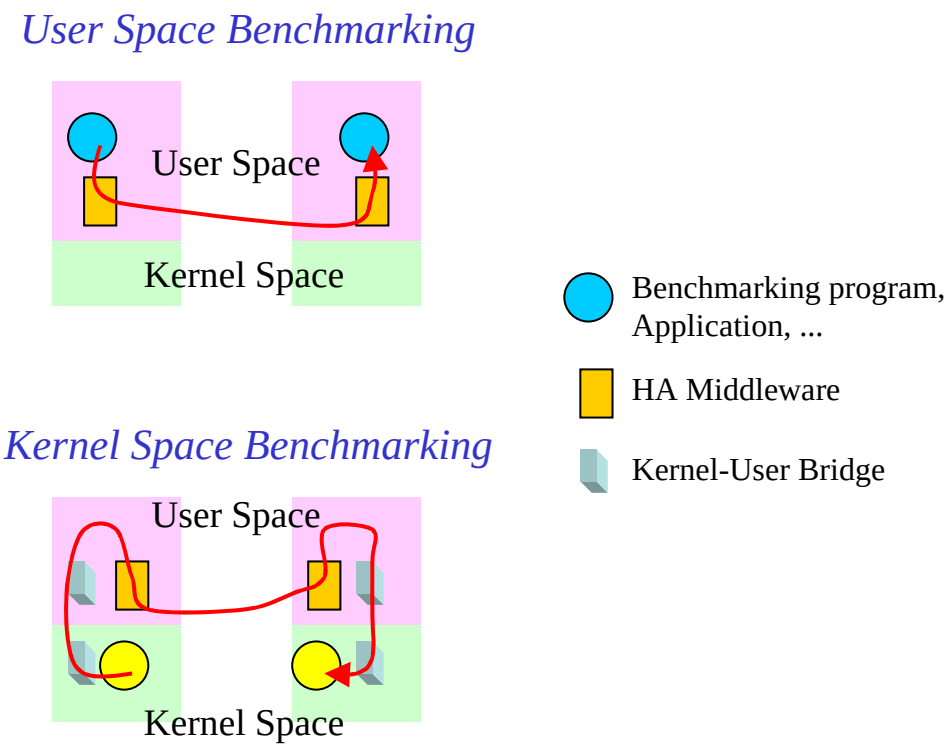


圖十七、Novel K-U bridge schemes for HA middleware 之運作架構

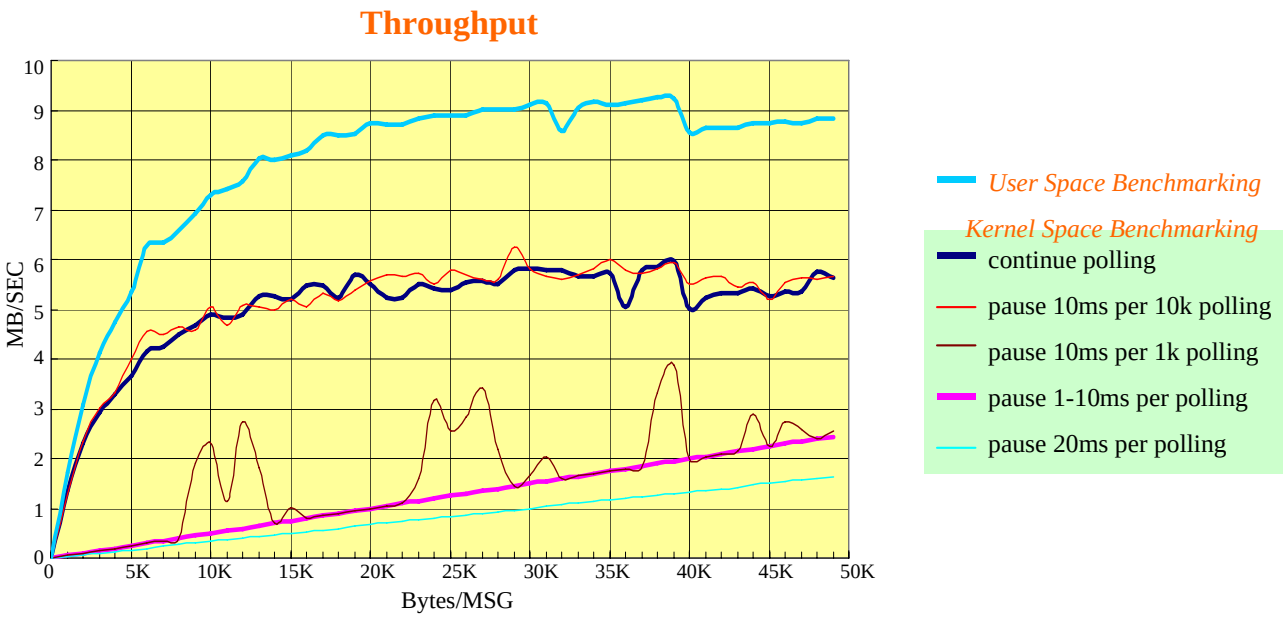


圖十八、Checkpoint-Based Benchmarking 測試示意圖

利用 checkpoint 功能將資料以 reliable 傳遞至各 Cluster Nodes，以量測其所花費之時間。共計針對不同的 nodes 數及"user relay manager" 方式之組合，去量測其資料傳輸速度，並計算出 Throughput 資料流量(Mbytes Per Sec)，以觀察、分析其效能；並針對 Linux-based 之 User-Space 及 Kernel-Space 之應用分別設計 Benchmark 測試程式，以瞭解在 Novel K-U bridge schemes for HA middleware 之運作效能，如圖十九所示。



圖十九、Kernel-Space vs. User-Space Benchmarking



圖二十、Novel K-U bridge schemes for HA middleware 效能測試－Throughput for 2 Nodes

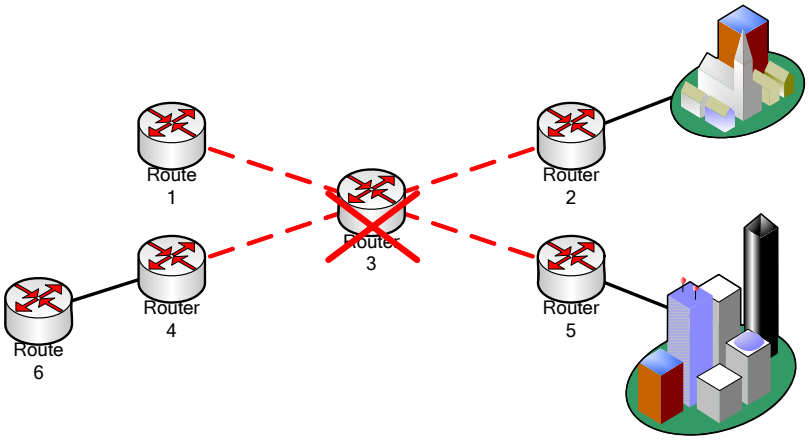
在以下硬體環境量測所得的效能結果，如圖二十所示。

- PC ： 2 nodes
- CPU ： x86 platform, Celeron 2.6G
- Network ： 100Mbits/s , private LAN

分析效能結論：在 User-Space 之效能頗佳，資料傳送速度可近於 wire-speed；在 Kernel-Space 應用測試中，以 Continue Polling 及 Pause 10ms per 10k polling 等二種方式的 Kernel-User Bridge 效能較佳；Kernel-Space 之效能較 User-Space 略為降低，約為 User-Space 之 60%左右。所以本 Novel K-U bridge schemes for HA middleware 的架構設計之效能，將可符合 Carrier-Grade 之需求。

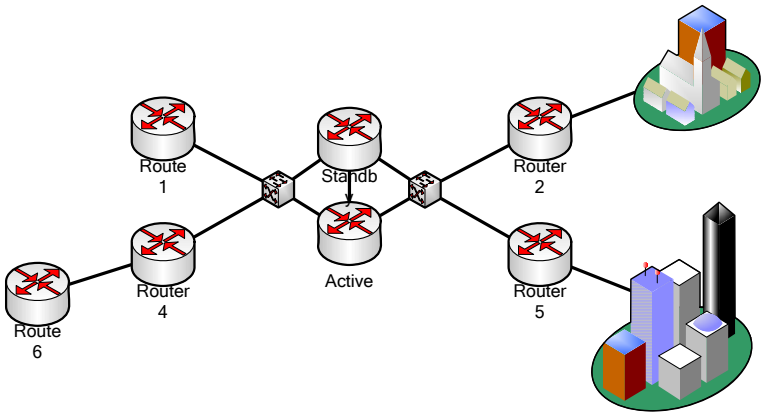
3.3.2 Fast recovery mechanisms of routing paths for HA middleware 的架構設計與效能分析

當網路中 Router 中斷服務，一般做法是利用 Redundant Router 來接手封包轉送之工作。若是 OSPF 則利用 Graceful Restart 方式避免路由資訊重新計算，如圖二十一所示以 OSPF 為例，圖中所有路由節點皆支援 Graceful Restart，因此當 Router 3 無法繼續服務，則 Router 2 的鄰居路由節點將進入 Helper 模式進行 Graceful Restart，但是此方式所造成服務中斷時間往往太長，而無法達到寬頻電信等級之標準。



圖二十一、網路中路由節點中斷服務

為了避免 Router 進行備援時，網路服務中斷時間過長，我們提出了一套快速路由路徑恢復機制，架構如圖二十二所示：



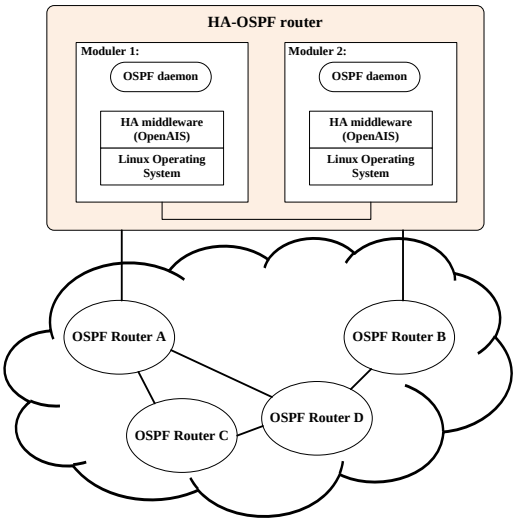
圖二十二、快速路由路徑恢復機制架構

在網路中加入了另外一個 HA-OSPF router，如圖二十三所示，在此 router 中包含了 2 個 router 模組，分別為 active 與 standby，在同一時間之內只會有一個 router 模組進行封包轉送服務，並且與其他 router 進行控制訊息的交換

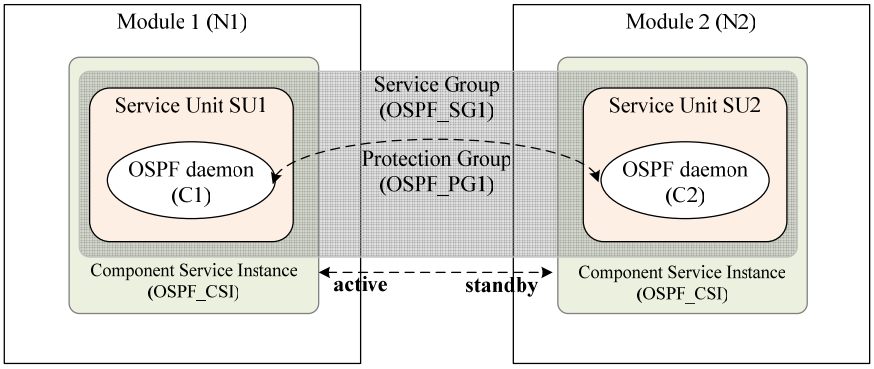
我們將利用 HA Middleware Generic Functionality 所提供之 AMF 來提供 Heartbeat 服務，如圖二十四所示，我們將利用 2N 方式進行備援，我們定義了 OSPF_SG1 這一個 Service Group 以及 OSPF_PG1 這一個 Protection Group，OpenAIS 會根據此二定義來決定哪一個模組是 Active 或是 Standby 來進行備援。

Active 會定期送 heartbeat 給 standby，standby 可以根據 heartbeat 得知 active 現在仍在服務，當 standby 沒有接收到 heartbeat 之後便知道 active 中斷服務，此時 standby 可以接手原本 active 的封包轉送工作，且此時 standby 的狀態也會變成 active。

本計畫利用工研院所設計定義的 API，其中 iclhaInit()用來初始化 AMF 以及 Checkpoint 相關設定，iclhaGetState()可以在程式中快速取得程式的身分，如 active 或是 standby，以方便設計相關程式流程，iclhaWrite()以及 iclhaRead()是用來對 checkpoint 進行寫入以及讀取的動作，這一些 API 可以幫助我們在開發的時候更容易進行。



圖二十三： HA-OSPF router system design



圖二十四: HA-OSPF router with SAF 2N Model

圖二十五是 Active 與 Standby 執行時的 pseudo code，當 router 模組的身分為 Standby 的時候，會從 checkpoint 讀出 routing table，再將 routing table 備援寫入系統中，若 router 模組是 Active 的時候分為 2 種情形，若先前狀態是 standby，則代表是剛接手工作，所以必須先執行 ospf 路由程式，並且立刻發送 Hello 以及 Link State Request 訊息給 neighbor router，否則就定期將 routing table 寫入 checkpoint 供 standby 進行備援動作。

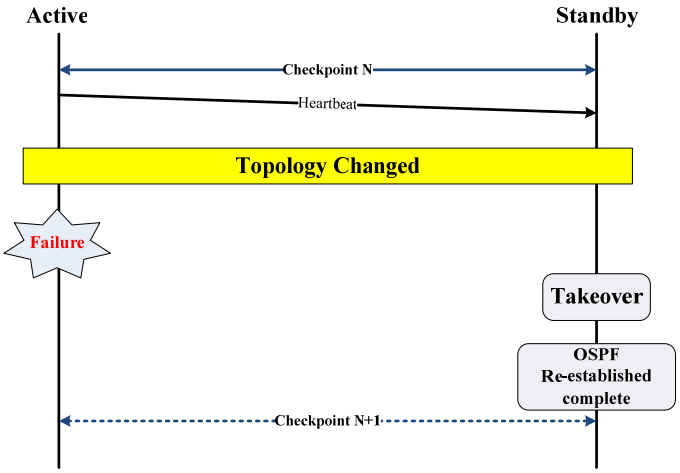
```
void callback()
{
    iclhaGetState();

    if(haState is Active) {
        if(previous haState is Standby) {   //takeover occurred
            start the OSPF routring procedure;
            send the OSPF Hello and Link State Request message;
        }
        else {
            write the routing information to the checkpoint;
        }
    }
    if(haState is Standby) {
        read the routing information from the checkpoint;
        update the information to its routing table
    }
```

```
}
```

圖二十五: Callback function pseudo code

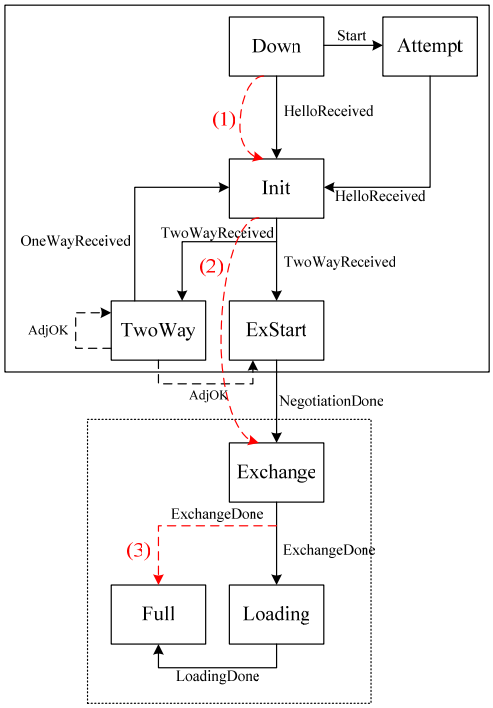
如圖二十六所示，在換手的過程中網路拓蹼可能會有所變更，因為 standby 在接手之前所得到的路由資訊並不包含拓蹼改變後的資訊，因此若不處理，資料封包將可能無法被轉送至正確的網路區段，因此我們提出了 OSPF 狀態快速轉換機制用來加快更新後的路由資訊。



圖二十六：Topology changed between checkpoints

如圖二十七為 OSPF 狀態轉換圖，我們將加快狀態轉換的流程，讓 router 可以快速進入 Full 狀態，作法如下：

1. 當 router 模組接手工作之後，一開始狀態為 Down，router 將立刻送出 Hello 訊息並將狀態設定成為 Init
2. 我們將略過 ExStart 狀態讓 router 直接進入 Exchange 狀態，準備交換 link state 封包
3. 在 Exchange 狀態 router 將會送出 link state request 訊息，向其每一個 neighbor router 要求最新的路由資訊，並且進入 Full 狀態，在此狀態下，再根據先前取得的 link state 資訊求得最新的 routing information，此時 router 便有最新的網路拓蹼的路由資訊

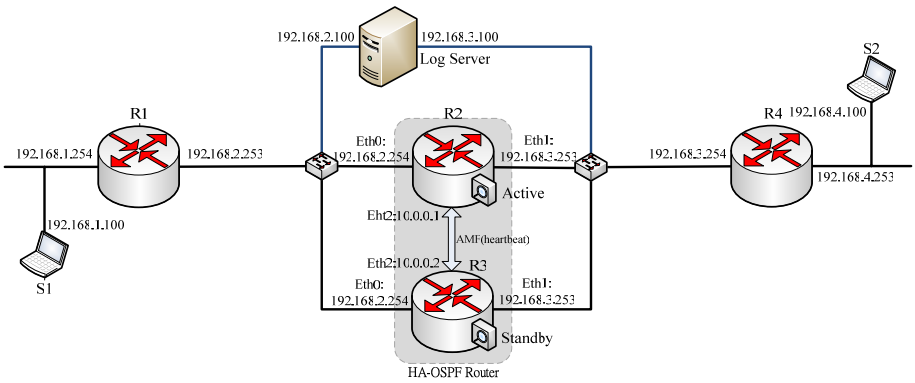


圖二十七：The state diagram of the proposed mechanism

如圖十五所示，當 active 在轉送封包的時候，active 也會將目前 active 上的路由資訊透過 K-U Bridge 介面取得再利用 HA Middleware Generic Functionality 的 checkpoint 傳送給 standby，以確保 standby 會收到來自於 active 最新的路由資訊。當 standby 接收到此路由資訊，standby 再透過 K-U Bridge 將此路由資訊更新至 kernel space 的 routing table，因此 standby 可以將自己本身的路由資訊更新成為與 active 一樣，故 standby 上將擁有與 active 一樣之路由表資訊，當 standby 接手封包轉送之工作，可以知道如何將封包轉送至正確的網路位置。

而在實際測試系統架構方面如圖二十八所示，其中 R1、R4、以及 R5 為一般的路由器，而 R2 以及 R3 為我們所設計有包含 HA Middleware Generic Functionality 以及設計有 K-U Bridge 的路由器，支援 checkpoint 與 AMF 服務，R2 與 R3 之間有一連結，所有 AMF 以及 checkpoint 資料皆由此介面傳遞。

在此圖中，R2 與 R3 連接二個網路，負責將封包轉交至正確的網路位置，且 R2 與 R3 皆配置相同數目之網路介面並給予相同之裝置名稱，如此在進行備援動作時才不會因為裝置不同而導致路由資訊無效。Log Server 主要負責紀錄網路上所有端點發生的事件，例如 R6 送資料給 R7 之前會先將此動作紀錄在 Log Server，R7 收到資料之後會把接收到資料的動作紀錄在 Log Server，如此可得知當 Router 中斷服務之後，需要多久時間才能恢復網路連線。



圖二十八：實際測試架構圖

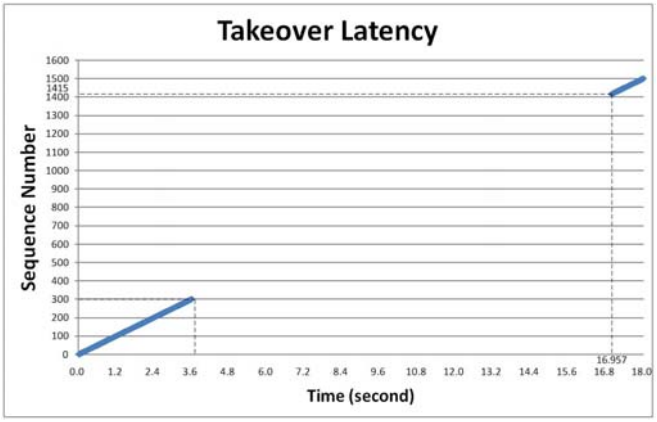
3.4 結果與討論

本研究第一年度首先以 OSPF 為例做實際測試，先比較沒有備援路由資訊與有備援路由資訊以及利用快速狀態轉換的備援方式在基礎網路恢復連線所需的時間，如表一所示，此表為根據上述三種方法所測得的網路恢復時間，我們取出第 10 次測試結果畫成圖二十九。

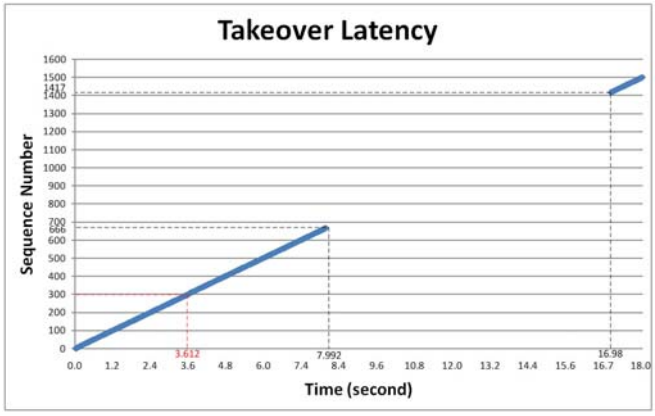
表一: Takeover latency (sec)

No	Scenario 1	Scenario 2	Scenario 3
1	13.369	10.001	3.253
2	13.357	8.880	3.240
3	13.927	8.988	3.252
4	13.369	4.008	2.496
5	18.361	8.988	3.240
6	13.356	14.989	2.577
7	12.964	8.989	3.240
8	13.357	8.996	3.241
9	13.344	8.989	3.232
10	13.357	8.988	3.240
Average	13.876	9.182	3.101

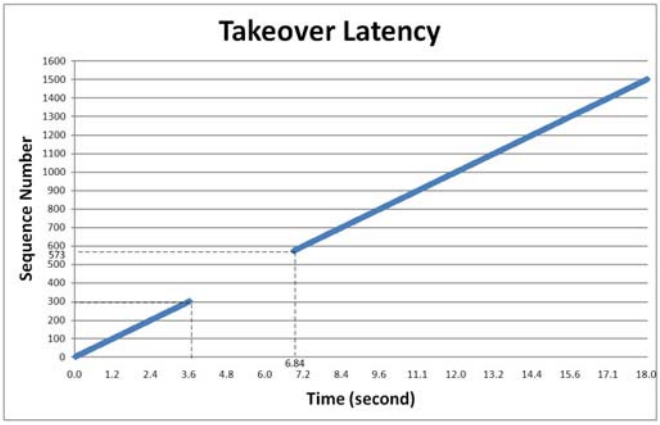
如圖二十九所示，分別測試此三種方法，根據結果顯示沒有備援路由資訊(Scenario 1)的方法，網路恢復時間約為 13.876 秒，備援路由路徑資訊(Scenario 2)的方法網路恢復時間約為 9.182 秒，而快速狀態轉換以及備援路由路徑(Scenario 3)的網路恢復時間約為 3.101 秒



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

圖二十九：Trace of packets received at Log Server

但因為上述之 routing table 備份皆屬於 user space 的動作，無法將正確的路由資訊更新至 kernel 的 routing table，且因為各個 routing protocol 設計之 update timer 皆不同，因此並無法立即得知 routing table 有更新。因此在本研究第二年度所提出的 Routing Table Manager redundancy for HA Middleware 架構，application 可透過此機制立即將獲得之 routing 資訊更新至 kernel space 的 routing table，故若 active 發生中斷，當 backup router 接手之後，也能夠使 routing protocol 重新更新路徑資訊之前，透過之前所更新的 routing table 資訊將封包轉送至正確的網路區段；目前在第一年 Fast recovery mechanisms of routing paths for HA middleware 的成果中單以 OSPF 做初步測試驗證，如圖二十九所示平均中斷恢復時間可在 420 ms 內完成。預期第二年度當所有路由服務(如 RIP 或 BGP 等)整合至系統時，整體中斷恢復時間可在 1 秒內完成，將可達成與國際大廠如 Cisco 等電信等級路由器設備相當的效能。

第 4 章 TCP takeover designs in HA middleware

本研究提出 Multiple TCP connections takeover in HA middleware 的創新架構設計擴充在圖九中 HA Middleware Enhanced Functionality 的 HA-Compliant L3/L4 Service 所需功能並提昇其效能；本研究主題的內容分別說明如下：

4.1 研究背景及文獻探討

A. 背景

隨著科技的進步，人們在日常生活中也越來越仰賴於網路上所提供服務。因此，網路服務提供者必須要能提供高可用性(high availability)的服務(例如，99.999%的正常運作時間)以滿足使用者的需求。然而，從使用者的觀點來看，高可用性並不意味著服務的連續性(continuity)。當服務必須進行容錯移轉(failover)時，維持服務的連續性代表著讓使用者無所查覺以及以最少量的效能降低達成容錯移轉。

現今日各式各樣的網路服務中，許多更是需要穩定的 TCP 連線才能達到電信等級的服務品質，例如 telnet、world wide web (www)、ftp、MOD、email 等。因此，如何能在可能遭遇 failover 的情況下，迅速地接管並恢復 TCP 連線是一個重要的課題。

此報告總結了我們第一年的研究成果。我們將描述如何能提供應用程式高可用性、高連續性的 TCP 連線。我們探討了一些可能會遭遇的問題以及使用了 ns2 網路模擬器來進行效能的評估，另外我們也使用了 TCPCP2 以及 OpenAIS 來進行實際的程式的開發來評估其效能。

B. 相關研究與系統架構

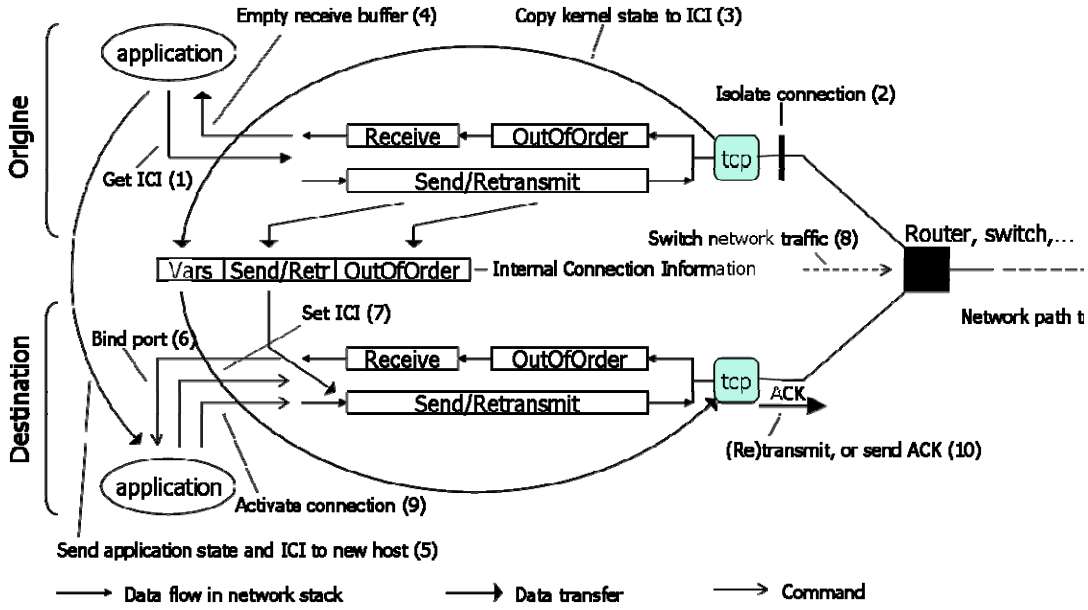
B.1. TCPCP/TCPCP2 與 OpenAIS

當正在服務中的行程所在的主機發生故障時，為了能繼續提供使用者服務，我們勢必要將該行程轉移至另外一台主機上。這樣的動作通常需要以下的基本元素:

1. 將應用程式狀態轉移至新的主機。應用程式狀態可能包含檔案名稱、目前鎖在檔案中的位置等。
2. 確保送往原本連線的封包能傳送至新的主機。
3. 新的主機能恢復原有的連線，與使用者繼續溝通。

在達成 TCP 連線的轉移的方法中，有的需要在 client 端與 server 端使用專用的協定，如 Mosix[1]及 MIGSOCK[2]，有些則不需要 client 端的修改，如 TCPCP /TCPCP2[3][4]。本研究重點是希望能在 client 端不需進行程式修改的情況下，達成 TCP 連線的轉移，因此將採用 TCPCP/TCPCP2 作為主要連線轉移的工具。以下簡介其基本運作機制。

TCPCP/TCPCP2 需要修改作業系統中的 TCP 和 socket 實作。應用程式可以藉由所提供的 API 來存取 TCP 連線的 kernel state（TCPCP 中稱之為 Internal Connection Information，ICI；TCPCP2 中稱之為 Socket Information，SI。此報告統稱為 ICI）。此 ICI 可藉由其他機制傳送給另一台主機，然後該主機再將 ICI 設回至其 kernel，此時便完成了 TCP 連線的轉移。TCPCP/TCPCP2 的開發尚未完整[5][6]，不過卻提供一個達成網路應用程式高可用性服務的開端。圖三十轉移一個連線的細節程序。

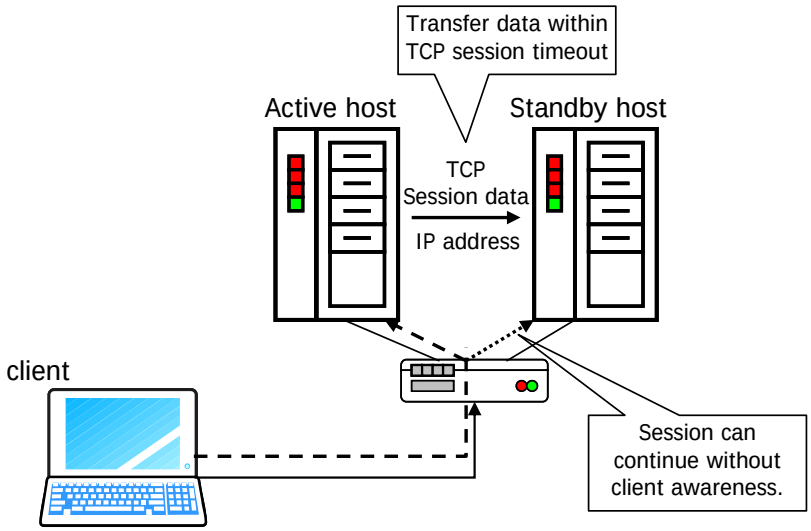


圖三十、tcpcp 轉移一個連線的細節程序

OpenAIS 是一個將 SA Forum 所制定的 Application Interface Specification （AIS）標準實現的 Linux 中介軟體（middleware）。在[7]中，作者描述了如何利用 AIS 中的 Application Management Framwork（AMF）及 Checkpoint Service 來開發高可得性的應用程式。

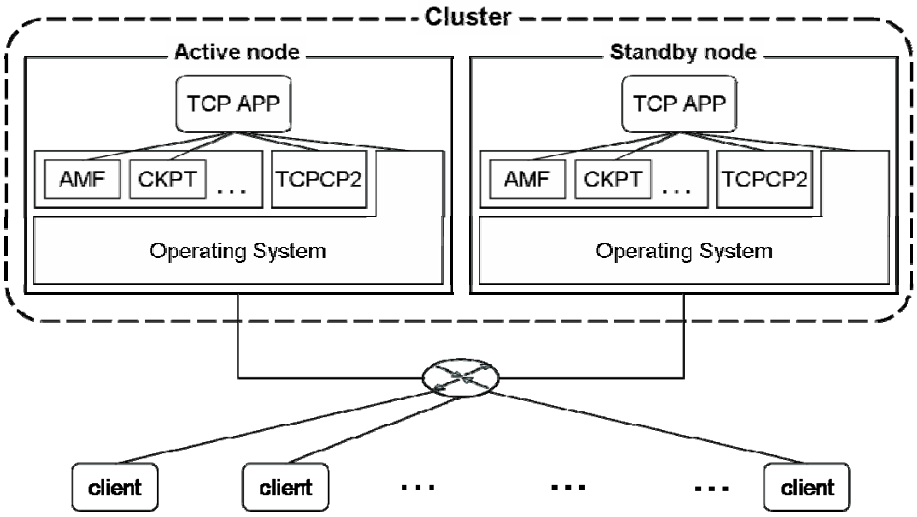
B.2. 系統設計

TCPCP/TCPCP2 並沒有提供將 ICI 傳送至其他主機的機制，我們的實作方法是結合了 TCPCP/TCPCP2 與 OpenAIS 中的 AMF 與 Checkpoint Service 來達成目標。基本的概念是將提供服務的 server（稱為 active server）中的 TCP 狀態同步的備分至另一台備援用的 server（稱為 standby server）。當 active server 故障時，standby serve 就能取而代之繼續服務，如圖三十一所示。



圖三十一、Active-standby servers 架構下的 TCP 連線接管方式

設計的架構如圖三十二所示。



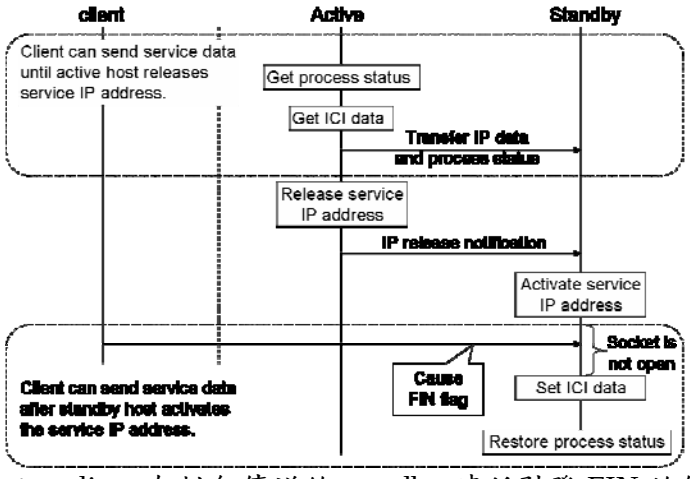
圖三十二、應用程式設計架構.

在程式正常運作時,active server 將應用程式狀態與 TCP 內部連線狀態週期性地傳送至其他 standby server。當 active server 故障時，AMF 會偵測到並且指定某一 standby server 成為 active server。這個新的 active server 可以利用最近一次的 checkpoint 資料來恢復應用程式及 TCP 連線的狀態，使其能繼續提供服務。在此研究中我們並未描述如何將流向原本 active server 的 IP 封包轉為流向取而代之的 standby server。在[5][8]中可以提供一些參考。

4.2 研究內容

C1. 由 FIN 所引發的 TCP 連線中斷

當 TCPCP 運作行程正常或不正常中斷時，該行程所使用的 descriptor 都將會被關閉，使得該行程發送 FIN 到 client 端來結束連線。一旦 client 端結束連線，standby 便無法再恢復連線。另外假如在當 standby 啟用原 active 的 IP 與設定 ICI 至 kernel 這段期間，client 有封包傳送給 standby（如圖三十三），此時也會引發 FIN 的傳送。解決方法是在 failover 發生時啟動 filtering 機制，使得 FIN 無法傳送到 client 端，此時便可順利恢復 client 與 standby 之間的連線。實現上可藉由行程不正常中斷時，接收作業系統所發出的 signal，並在 signal handler 裡啟動 filter 的機制。



圖三十三、client 有封包傳送給 standby 時所引發 FIN 的傳送情形

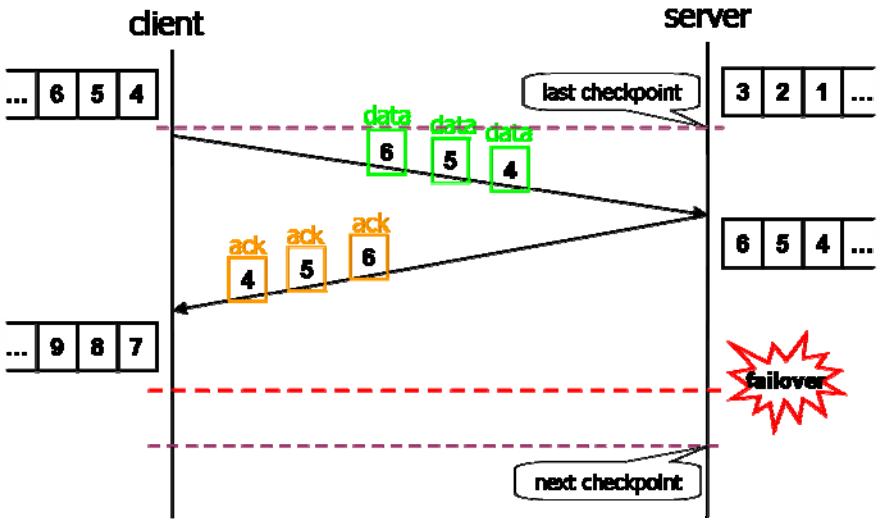
C2. active 與 standby 之間 TCP 連線狀態的衝突

由於 HA Middleware Generic Functionality 中的 checkpoint 是間歇性地運作且 failover 的

狀況大都是不可預期的，所以從上次做 checkpoint 的時間點到 failover 發生的時間點之間，可能會使得 TCP 的 send/receive buffer 的資料或 TCP 連線狀態不一致的問題。以下將針對 server 接收來自 client 端資料的部份，以及 server 傳送至 client 端資料的部份，分別討論其可能發生的問題以及可能的解決方法。此外尚有一些提高效能的可能作法。

- server 接收自 client 的資料

如圖三十四所示，在上次 checkpoint 至 failover 的期間，client 可能已經從其 send buffer 送出一些資料、server 可能已經回了 acks 他所收到的資料、而且 server application 可能已經從其 receive buffer 擷取出了資料。此時，當 standby server 嘗試利用上一次的 checkpoint 資料來恢復服務時，client 的 send buffer 此時已無法再重送稍早之前的資料了（因為已經被 acknowledged 了）。



圖三十四、當 failover 發生在 checkpoint 之間時所會產稱的問題

此狀況在某些應用服務下是可以容忍的。例如 SSH/Telnet、SIP 或其它強調即時性的連線，舊資料皆可能不會影響之後的運作，因此可以丟棄。此類的連線在服務轉移到 standby server 之後，只須利用來自 client 的封包，將 sequence number 等 TCP 連線的資訊更新即可。

而對於強烈要求資料一致性的服務，我們希望可以延遲 server 端 ack 的傳送，直到下次 checkpoint 的時候，才傳遞自上次 checkpoint 到此次 checkpoint 間收到的資料的 ack，藉此讓 client 有機會重傳因 active server 發生錯誤而遺失的資料。但此方法可能會造成效能的降低，必須設法將影響降至最低。例如盡可能在 client 傳送 congestion window 大小的資料量之前，就執行 checkpoint，並回傳 ack 給 client，當 client 收到 ack 後，就可持續傳遞新的資料，藉此保持 client 的傳送速度。此外，延遲 server 端 ack 的傳送將會造成 client 對於 round-trip time 的誤判，亦需設法排除。

- server 傳送資料至 client

當 active server 發生錯誤而將服務轉移到 standby server 後，standby server 會重新傳送舊資料，若碰巧舊資料的 sequence number 與 client 欲接收的新資料的 sequence number 相同，而將舊資料誤以為新資料，並將其接收。此情形稱為 sequence number wrapped。

要防止此情形發生的第一種方法是：必須確保兩次 checkpoint 的時間之間，不會傳送超過總 sequence number 一半大小的資料量，即 2^{31}bytes。因為若 server 連續送出 sequence number 0~2^{31}(共 2^{31}+1 bytes)的資料後，仍無執行 checkpoint，此時發生 failover 的話，由於 client 預期接收的資料的 sequence number 已變更為 2^{31}+1~2^{32}-1 以及 0 (共 2^{31}bytes)，而 standby server 退回上次 checkpoint，重新傳送已傳送過的 sequence number 為 0 的舊資料，此資料卻被 client 誤認為新資料而將其接收，造成 sequence number wrapped。故須限定兩次

checkpoint 的時間內，不能傳送超過 2^{31}bytes 的資料量，以排除此問題。

第二種方法則是使用 Protect Against Wrapped Sequence number(PAWS)機制，也就是在 packet 中加入 timestamp。client 除了確保新資料的 sequence number 沒錯以外，還須確保新資料的 timestamp 比舊資料的 timestamp 新才會將其接收，藉此避免 sequence number wrapped 的問題。利用此方法的話，則兩次 checkpoint 的時間內就沒有不得傳送超過 2^{31}bytesr 資料量的限制，藉此可以放寬 checkpoint 的週期，降低 server 花在執行 checkpoint 上的負擔。但若使用此機制，則必須將 active server 與 standby server 的時間同步才行，以免 standby server 重傳資料時，卻因 timestamp 的不同步，導致 client 誤將舊資料當成新資料接收，或將新資料當成舊資料丟棄。

C3. active 與 standby 之間同步資料量

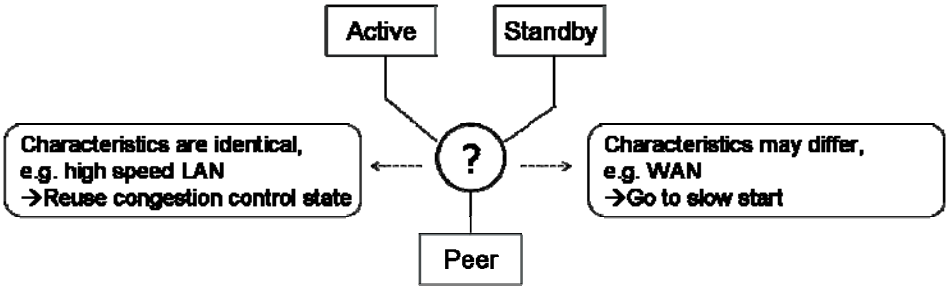
TCP 連線的 send/receive buffer，佔了 active 與 standby 之間連線狀態同步的絕大部份。在一擁有大量連線的 server 中執行 checkpoint 時，大量的 TCP 連線狀態資料及 send/receive buffer 等資料須被記錄及同步到 standby server。隨著連線個數的增加，checkpoint 的執行對於系統的負擔也越重。為改善其效能，可將連線類型加以分類。對於不要求資料一致性的連線，如 SSH/Telnet、SIP 或其它強調資料即時性的連線，可不必記錄與同步其 send/receive buffer 的資料。當此類連線所佔比例高時，可大量減少平均每個連線需記錄及同步的資料量，改善執行 checkpoint 所需的負擔，提高最大的可連線數。另外，亦可調整 TCP 連線的 buffer size，以空間換取更多連線的存活率，如圖三十五所示。

C4. 新連線的壅塞控制（congestion control）

目前 TCPCP 對於在連線轉移時所面對的 congestion control 的方面，是採取高度保守的方法，即在連線轉移後所採取的 congestion control 都是由 slow-start 開始。

圖三十六表示當連線轉移到 standby 端時所可能採取的 congestion control state。若 standby server 端位於連線特徵已知的環境中，例如 active 與 standby 同在一個 LAN 中，則可延用連線轉移前的 congestion control state；反之，如果 server 端位於連線特徵未知的環境中，例如 active 與 standby 在 WAN 中，則進入 slow-start 的狀態，也就是從 congestion window = 1 可能會是比較保險的方法。

另外也可以嘗試採取 congestion control 中 fast recovery 的方法，將原本 congestion control state 中 congestion window 與 slow-start threshold 變成一半，或者直接延用原本的 congestion control state，如此就不用從 slow-start 開始。

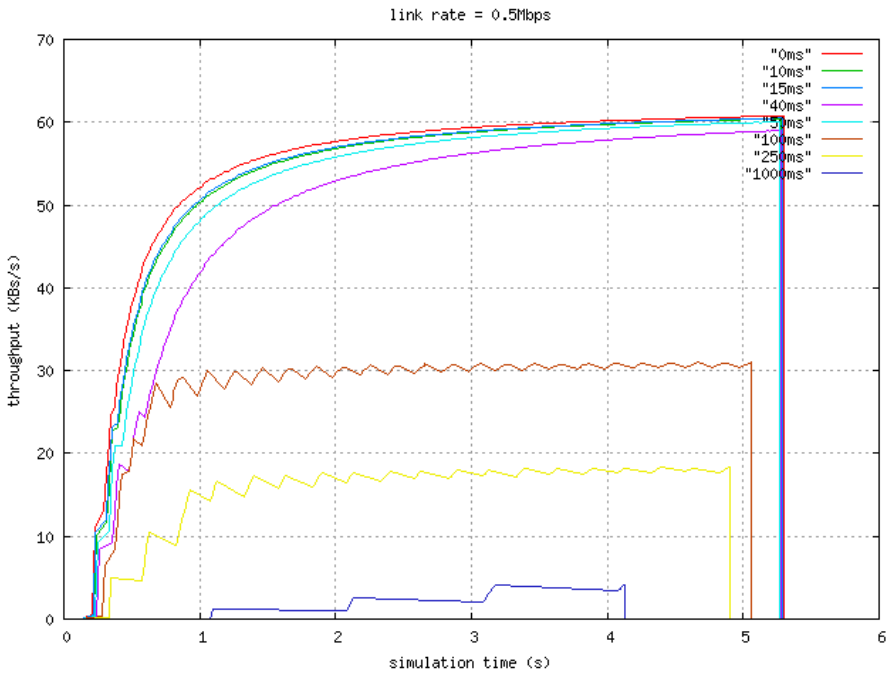


圖三十六、congestion control 於 connection parsing 的連線策略

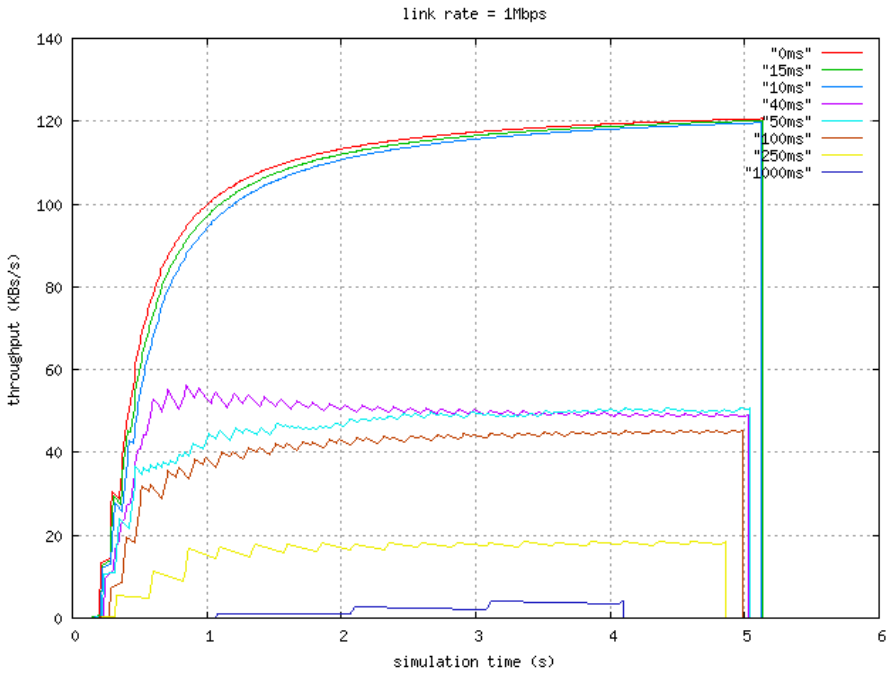
4.3 數據結果

D1. Server 端的 Delaying Ack

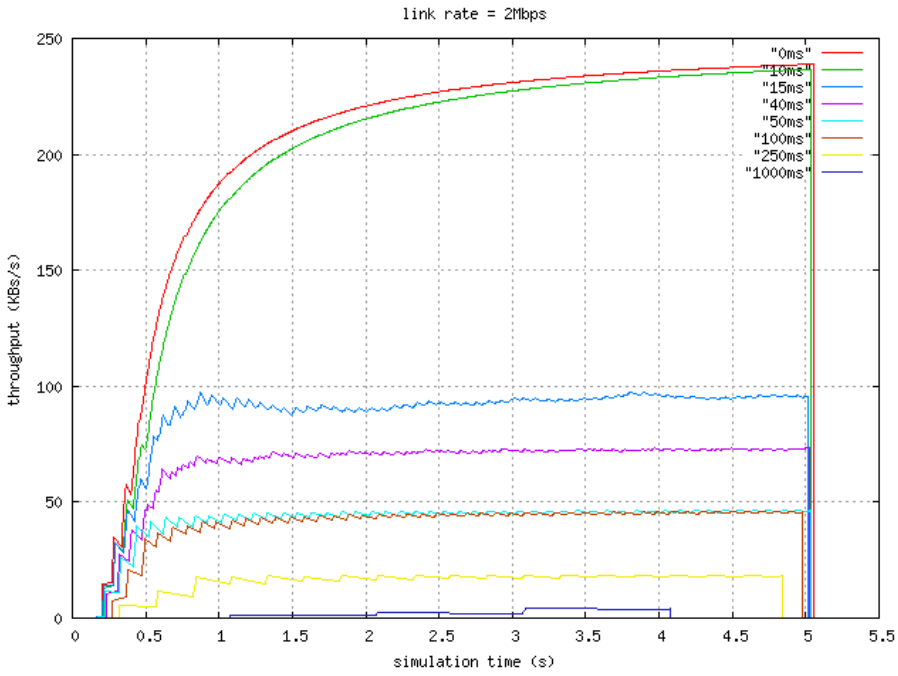
我們將 ns2 模擬器稍作修改來模擬在 D2 中所描述的延遲 ack 的方法。在拓樸中我們建立了一個 client node、一個 server node，以及一個負責幫 client 轉送封包至 server 的 router。Client node 不斷地送出 ftp 資料至 server node。模擬的結果如圖三十七所示，其中 link rate 分別為 0.5、1，以及 2Mbps。由圖可以看出，在 checkpoint interval 調增至某個程度時，throughput 開始有大量的遞減，而且至少降幅為沒有使用 checkpoint 時的一半以上。



圖三十七 a



圖三十七 b



圖三十七 c

圖三十七、不同 checkpoint interval 下的 throughput 比較。圖三十七 a、Link rate = 0.5 Mbps。圖三十七 b、Link rate = 1 Mbps。圖三十七 c、Link rate = 2 Mbps.

D2. Congestion window 預測

我們使用 ns2 模擬器進行比較下列恢復連線時，congestion window 所設的大小：

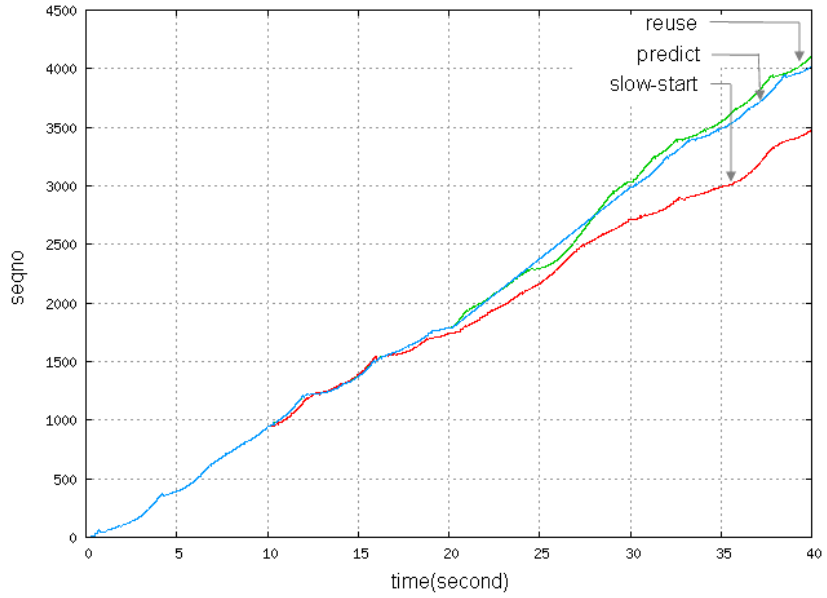
- **slow-start** 將 congestion window size 設為 1
- **reuse** 使用上次 checkpoint 裡的 congestion window size
- **predict** 使用良好預測的 congestion window size

在 predict 方法中，我們採用以下自[9]所衍伸出的公式：

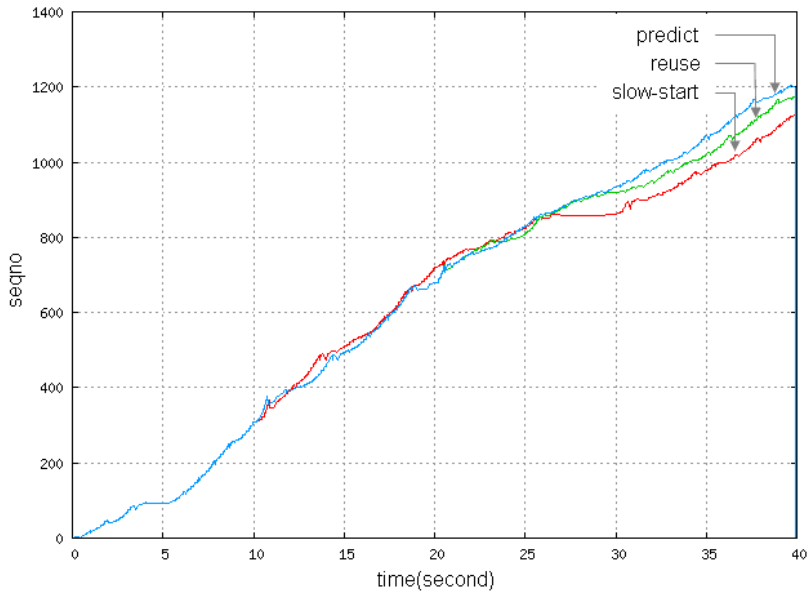
$$\text{Congestion Window Size}_{\text{predicted}} = 1.22/(RTT * sqrt(Loss))$$

模擬環境上，我們在 client 與 server 中間放置了一個 delay box。Delay box 代表 client 與 server 之間的網路。Link rate 設定為 2Mbps。Server 連續傳送了 40 秒的 ftp 資料至 client 端。在 10、20，以及 30 秒時傳送被中斷且馬上恢復連線。我們比較了在恢復狀態時分別採用 slow-start、reuse，以及 predict 時所導致的結果，其中 reuse 設定為 0.5 秒前的 congestion window 值。結果如圖三十八所示。

圖三十八 a 及圖三十八 b 分別為將 delay box 的 loss rate 設為 0~0.01 及 0~0.09。圖中 y 軸的 sequence number 代表的是傳送的 packet 個數。我們可以觀察到當 loss rate 較低時，slow-start 的 sequence number 明顯地成長得最慢；而當 loss rate 較高時，slow-start 與其他曲線的差距變得較小。另外我們也觀察到在 loss rate 高時，predict 比 reuse 來得更好。這是因為 loss rate 高表示網路狀況的不穩定，此時舊的 congestion window 已不能反應現在的網路狀況。



圖三十八 a



圖三十八 b

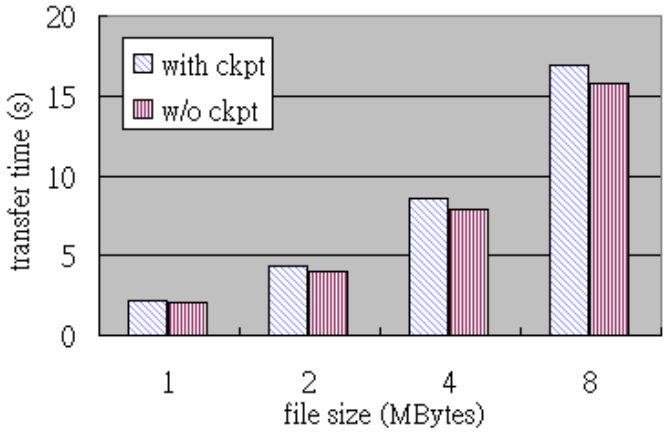
圖三十八、Sequence number 與 time 關係圖。圖三十八 a、Delay box loss rate = 0~0.01. 圖三十八 b、Delay box loss rate = 0~0.09.

D3. 應用程式開發及其效能測量

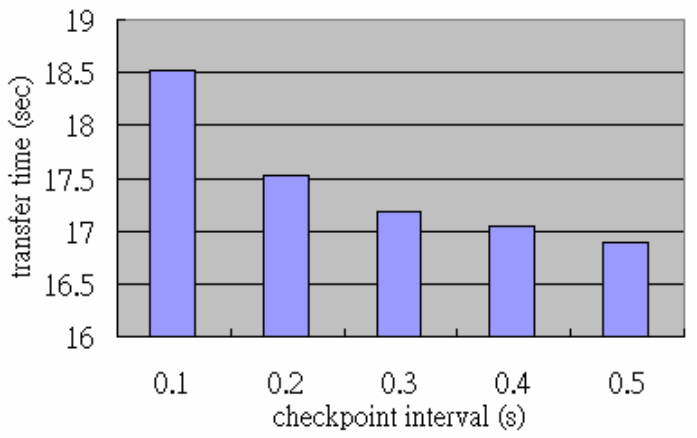
明顯地，在達成高可得高連續性與服務的效能之間必須取得一個平衡。Active 與 standby 之間的 checkpoint 更新越是頻繁，處理器真正花在服務上的時間就越少。

我們使用了 OpenAIS 以及 TCPCP2 開發了一個具 service availability 能力的簡易的檔案傳送程式。此程式在 client 與 server 建立連線之後就馬上傳送一個檔案至 client 端。

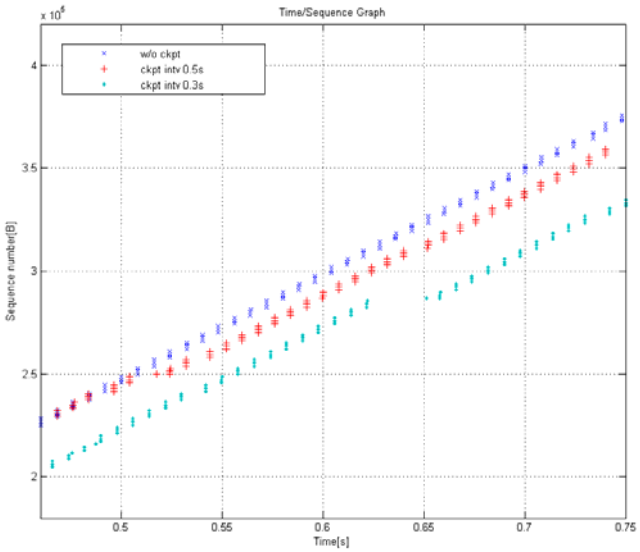
圖三十九比較了具 SA 能力以及無 SA 能力的程式在傳送不同檔案大小時所需的時間。結果顯示平均而言，具 SA 能力的程式需要多 6.3%的時間來傳送檔案。圖四十為傳送一 8MB 大小的檔案時，將 checkpoint interval 設為不同值時所需的傳送時間。可以看到在 checkpoint interval 由 0.1 秒調為 0.2 秒時，所需的傳送時間有大量的下降，而 0.2 以後下降趨於減緩。圖四十一比較在不做 checkpoint、checkpoint interval 為 0.5 秒，以及 checkpoint interval 為 0.3 秒時，server 端的 sequence number 成長圖。我們可以看到在 checkpoint interval 為 0.5 秒時，在 0.5~0.55 秒的地方 sequence number 有稍微停頓一下。類似的情況也發生在 0.6~0.65 秒時的 checkpoint interval 為 0.3 的曲線上。這是因為在那時候程式正在擷取 TCP 內部連線並且與 standby 更新。



圖三十九、傳送不同大小檔案所需的時間。Checkpoint interval 為 0.5 秒



圖四十、傳送一 8MB 的檔案所需的時間



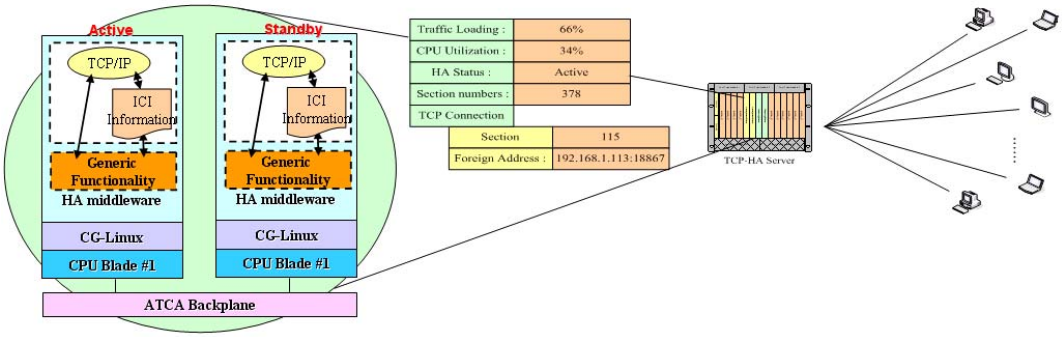
圖四十一、在 0.45 ~ 0.75 秒間的 sequence number 成長圖

4.4 結論

在此報告中，我們展示了我們第一年的研究成果。我們描述如何利 OpenAIS 以及 TCPCP/TCPCP2 來達到高可用性、高連續性的 TCP 連線。我們說明了如何藉由只有在 server 端程式已經擷取 TCP receive buffer 裡的資料以及作完 checkpoint 更新後才回送 ack 至 client 端，來避免當 failover 發生時 client 端的 send buffer 資料已經不存在的問題。模擬結果顯示 checkpoint interval 必須限定在特定範圍內，否則 throughput 會有嚴重的下降。我們也展示了在恢復 TCP 連線時，congestion window 的起始值是可以經過適當的預測與設定來達到更好的效能。我們也實際使用了 OpenAIS 以及 TCPCP 來開發了一個具備 service availability 能力的應用程式，實驗結果顯示有 service availability 能力的程式平均多耗費 6.3%的時間來傳送一個檔案。

綜合以上現有 TCPCP/TCPCP2 需改善的部分以及我們的研究成果，本研究期望能夠以 HA Middleware Generic Functionality 所提供的 AMF 及 Checkpoint Service 等功能為基礎，並整合第一年完成的 Novel K-U bridge schemes for HA middleware，擴充 HA Middleware Enhanced Functionality 以實現出可龐大連線個數、更短連線恢復時間、更穩定連線速度及管理監控介面等特點的 TCP 連線轉移解決方案，如圖四十二所示。本研究期望能將研究產出回饋業界與研究單位，其中包含了將改良後的 source code 貢獻至 open source projects (openAIS 或 TCPCP)、發表論文至國際研討會或期刊，並且申請專利。目前已將成果投稿

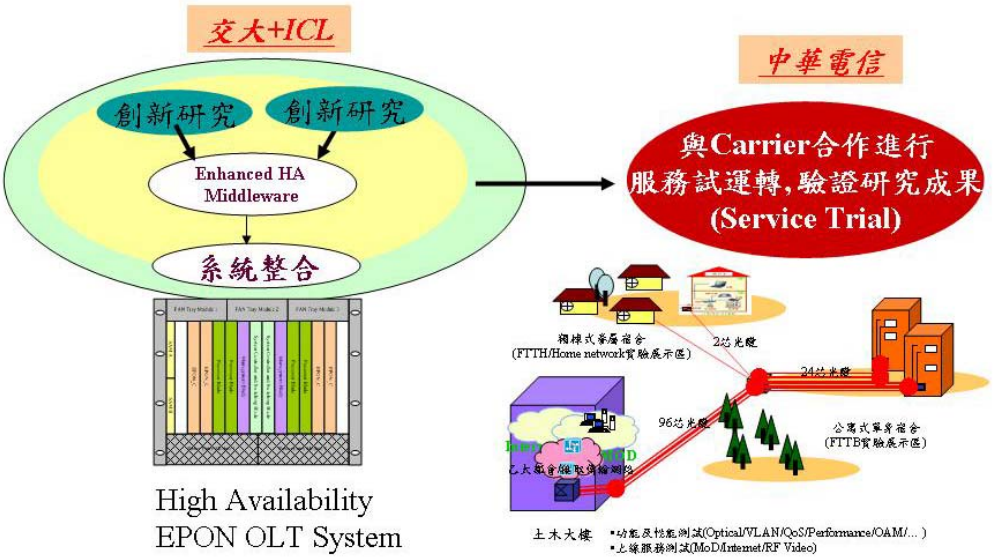
至國際期刊[10]。



圖四十二、高可得性、連續性的 TCP 連線轉移解決方案

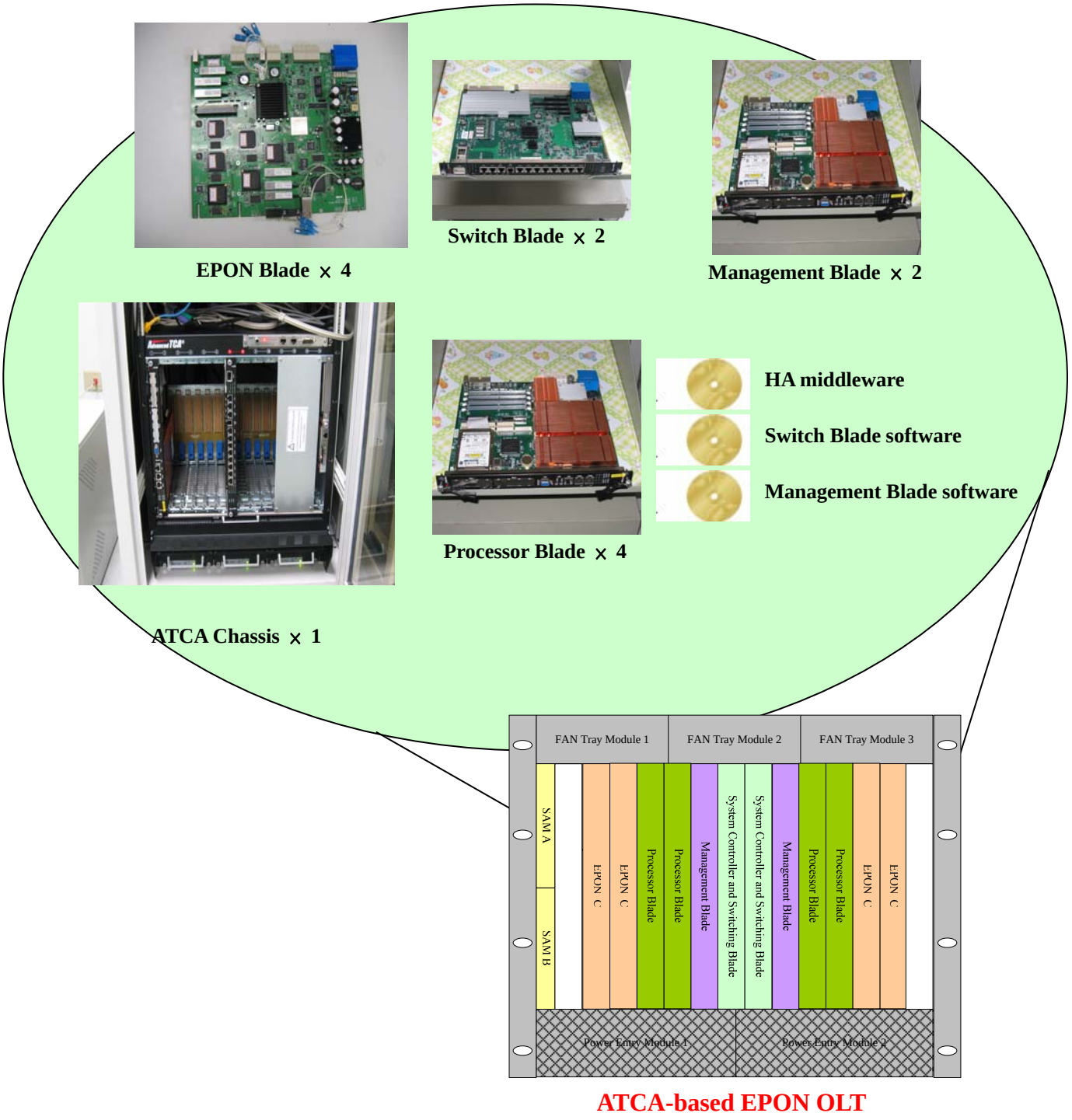
第 5 章 Carrier-Grade High Availability EPON OLT/Server 系統之建置與測試驗證

本計畫在執行方面將結合產學研各方面之能量共同完成，分工方式包括：(圖四十三)



圖四十三、計畫執行方式與分工概念圖 (ICL：工研院資通所)

- 採用業界或研究單位已完成或開發中的 ATCA 硬體平台與各式標準卡板、搭配資通所開發的 EPON 介面卡板與相關軟體來建置 EPON OLT 系統。其中標準卡板包含第一年度的兩套 Switch 卡板、兩套 Management 卡板及第二年度的四套 Processor 卡板；另外，四套符合 PICMG 3.0 規範的 EPON 卡板(第一年度及第二年度各兩套)。相關軟體包含有 Generic Functionality 的 HA middleware 與 Switch 卡板及 Management 卡板所需軟體，而完成如圖四十四所示的一個完整 EPON OLT ATCA-based Chassis 系統。



圖四十四、ATCA-based EPON OLT 系統

- 將與資通所共同整合 High Availability Middleware 的研究成果與 EPON OLT 系統的工作。將 K-U bridge、Multiple TCP connections takeover、High availability SIP 及 High availability Software Switch 等各項研究成果，與資通所的 EPON OLT 系統進行軟硬體整合，實際移植到開放平台標準的 EPON 設備上，並驗證其功能及效能。
- 將與資通所、中華電信研究所合作建置 EPON 系統可靠度測試環境，以現場服務運轉的角度，來驗證 High Availability EPON 系統與各項研究成果的效

能。本計劃的建置場地預計設在中華電信研究所內，並由資通所與中華電信研究所共同負責試運轉環境建置的規劃、試運轉環境現場測試、運轉狀況監控與統計工作、試運轉環境建置施工與維護、安排試用戶等工作。本計劃也將尋求與業界的合作，來取得所需的 ONU 及各項設備。

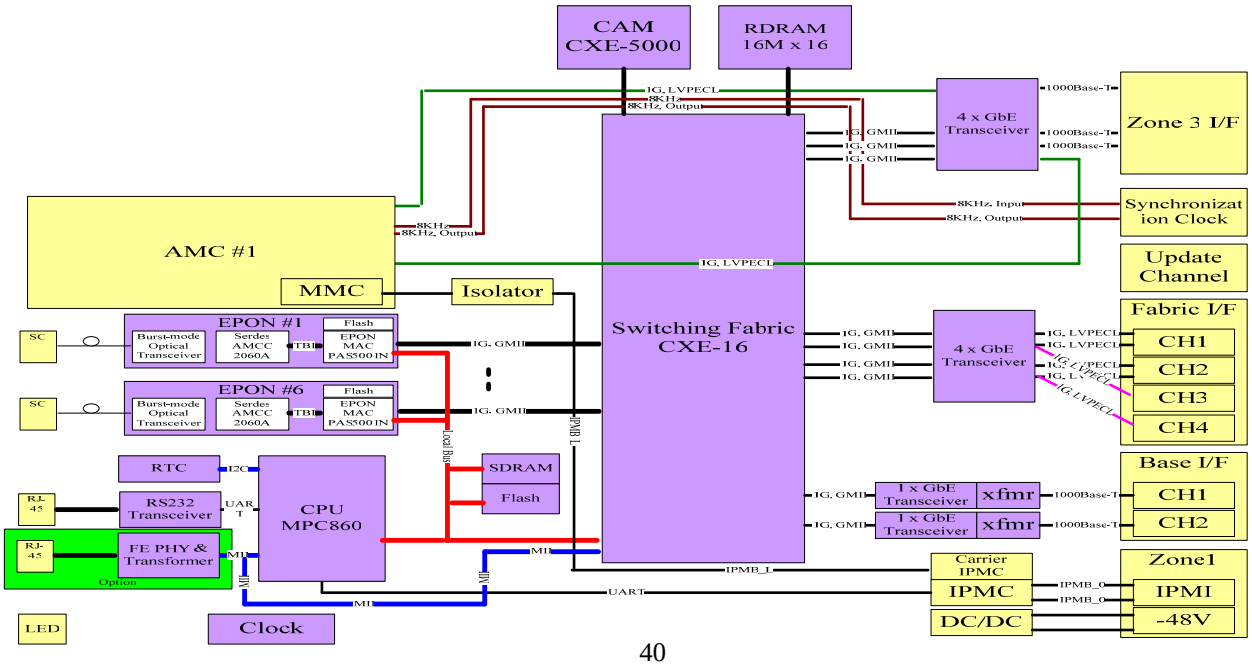
相關研究產出將回饋業界與研究單位，並協助改善可靠度的架構與機制，進而提升國內所開發的寬頻設備的可靠度。另外在 EPON 試運轉與可靠度測試的環境建置上，除了可供學界持續進行 EPON 相關研究外，也可作為發展各種網路應用服務的平台，了解各種應用服務在實際 EPON 網路的運作；或是作為廠商的 EPON 產品(包括 OLT、ONU)、或是 ATCA 硬體平台的各種卡板、各種軟體與 SAF middleware 標準互通與相容測試的平台，並尋求相關成果商品化的合作機會。

依據上述的計劃建置目標，第一年度已進行 EPON OLT 系統可靠度測試環境建置與相關測試流程規劃，基於電信等級開放平台架構(Carrier Grade Open Architecture；CGOA)，完成 EPON Platform 平台之 Middleware 與各模組單元的建置及單元測試，包括了 ATCA-based 之 EPON Blade、Management Blade、Switch Blade 等模組單元，並完成 CGOA EPON 平台之整合與其測試環境之建置，建立了一個完整的 EPON OLT ATCA-based Chassis 系統與環境。

5.1 EPON Blade 開發與建置

EPON OLT 卡板為整個 EPON OLT Chassis 系統之關鍵組件，為了支援 EPON OLT 功能之運作，進行了 EPON Blade 之設計與開發。整個 EPON Blade 的硬體設計如圖四十五所示，其並具有如下之特性：

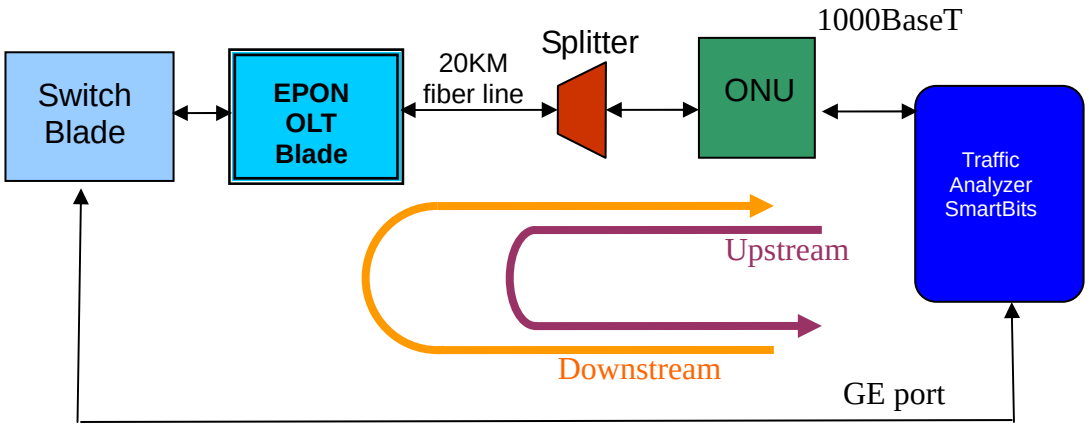
- 符合 IEEE 802.3ah 標準
- ATCA-compliant
- High Density: 6 PONs in a blade
- 支援 Flexible Fabric Interface
- 同時支援 Dual Star and Full Mesh Topology
- 支援 AMC Module 以作為未來擴增加強服務之用



圖四十五、EPON Blade 硬體功能方塊圖

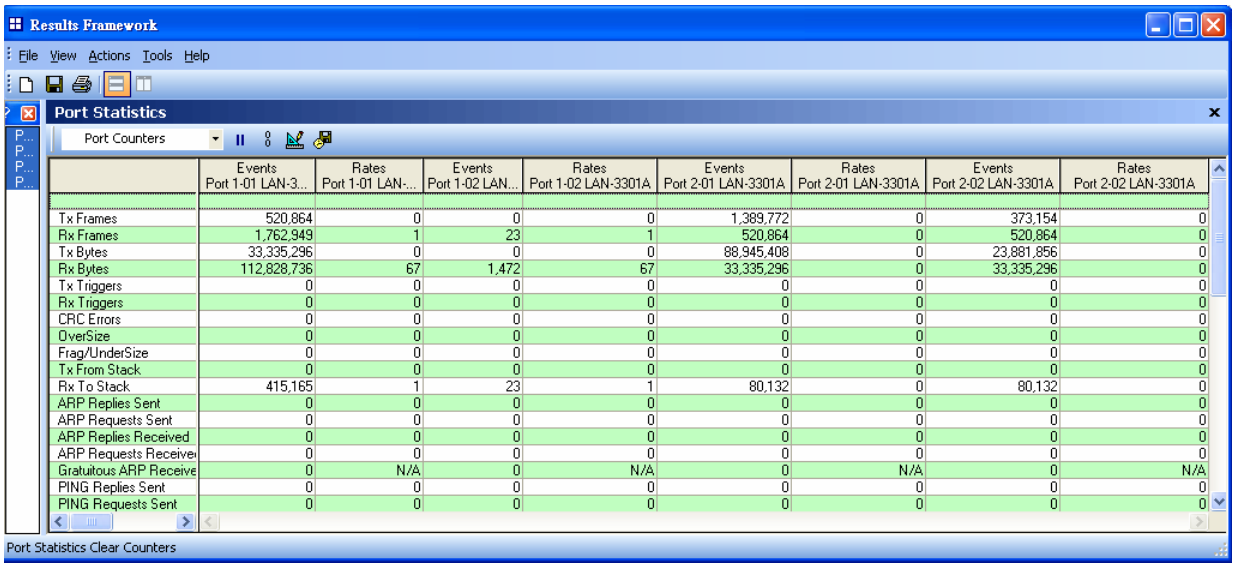
同時已完成 EPON Blade 之開發與建置，並就 EPON Blade 之資料傳輸功能進行驗證測試與效能分析。其測試架構如圖四十六所示，以 Switch blade 連結 EPON OLT blade，再連接至 ONU，而形成一資料路徑，並利用 SmartBit Traffic Analyzer 流量分析儀來傳送測試訊務(frame)，以檢視及驗證 EPON 網路之資料封包傳輸功能，其中包含了如下兩個方向的測試：

- Downstream：Traffic Analyzer SmartBits傳送frame → Switch Blade之front panel fabric port → EPON OLT blade → ONU → 流回至Analyzer SmartBits。
- Upstream：Traffic Analyzer SmartBits傳送frame → ONU → EPON OLT blade → Switch Blade & front panel fabric port → 流回至Analyzer SmartBits。



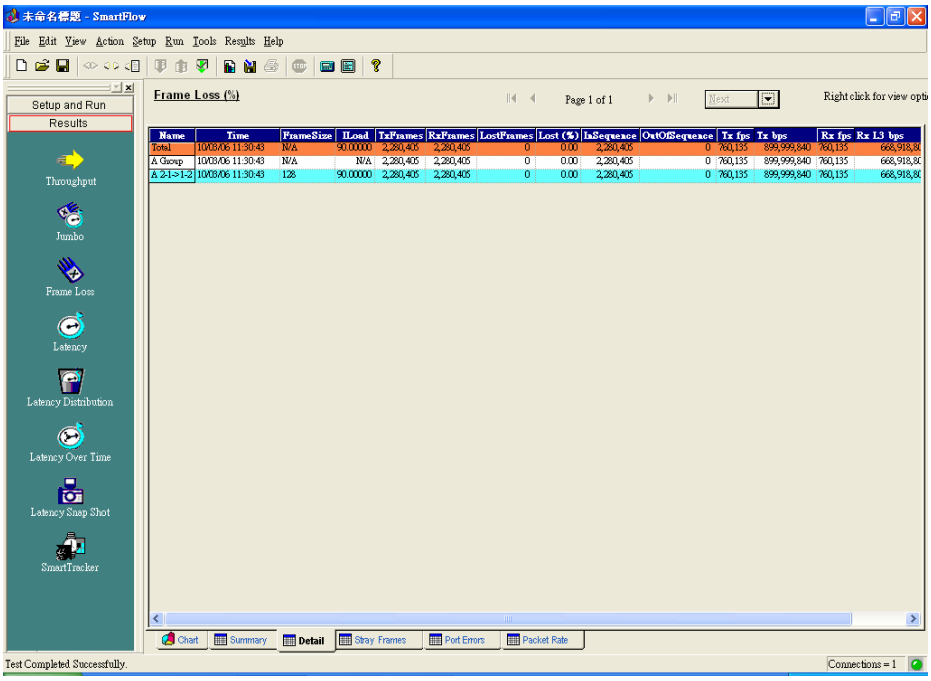
圖四十六、EPON網路測試架構

驗證結果如圖四十七所示，顯示資料的傳送與接收功能都能正常的運作。本計畫也進行了效能之評測與分析，驗證了 throughput 及 latency 等性能皆能符合規格需求。另外，也針對封包流失(packet loss)的情形加以量測，如圖四十八所示，結果顯示並無任何封包遺失。



	Events Port 1-01 LAN-3...	Rates Port 1-01 LAN-...	Events Port 1-02 LAN...	Rates Port 1-02 LAN-3301A	Events Port 2-01 LAN-3301A	Rates Port 2-01 LAN-3301A	Events Port 2-02 LAN-3301A	Rates Port 2-02 LAN-3301A
Tx Frames	520,864	0	0	0	1,389,772	0	373,154	0
Rx Frames	1,762,949	1	23	1	520,864	0	520,864	0
Tx Bytes	33,335,296	0	0	0	88,945,408	0	23,881,856	0
Rx Bytes	112,828,736	67	1,472	67	33,335,296	0	33,335,296	0
Tx Triggers	0	0	0	0	0	0	0	0
Rx Triggers	0	0	0	0	0	0	0	0
CRC Errors	0	0	0	0	0	0	0	0
OverSize	0	0	0	0	0	0	0	0
Frag/UnderSize	0	0	0	0	0	0	0	0
Tx From Stack	0	0	0	0	0	0	0	0
Rx To Stack	415,165	1	23	1	80,132	0	80,132	0
ARP Replies Sent	0	0	0	0	0	0	0	0
ARP Requests Sent	0	0	0	0	0	0	0	0
ARP Replies Received	0	0	0	0	0	0	0	0
ARP Requests Receive	0	0	0	0	0	0	0	0
Gratuitous ARP Receive	0	N/A	0	N/A	0	N/A	0	N/A
PING Replies Sent	0	0	0	0	0	0	0	0
PING Requests Sent	0	0	0	0	0	0	0	0

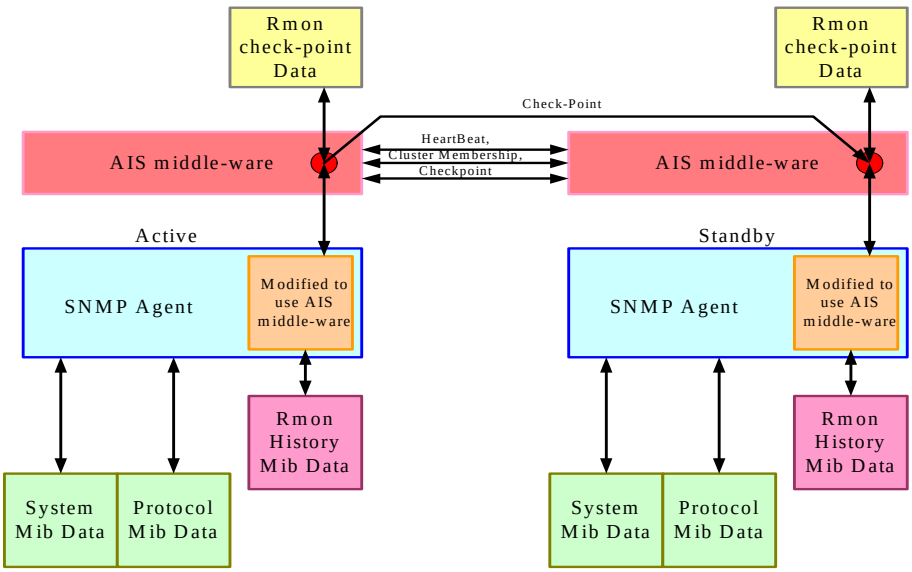
圖四十七、EPON 網路正常訊務測試圖



圖四十八、EPON 網路封包流失測試圖

5.2 Management Blade 建置與功能測試

Management Blade 負責整個 chassis 系統之管哩，提供了對 EPON OLT blade 與 Switch Blade 等單元之管理功能。在 management blade 上，建構了 Carrier Grade Linux 作業系統及前述所開發之高可用度中介軟體(High Availability Middleware)，以提供一個 reliable OS 與高可用度基礎建設(HA framework)的環境。在此架構基礎下，發展與建置了高可用性的管理軟體 SNMP agent，基本設計結構如圖四十九所示，提供可靠的管理服務。



圖四十九、高可用性管理軟體功能設計

針對兩套相互備援的 Management Blade 及所建置的 HA Middleware 與管理軟體，

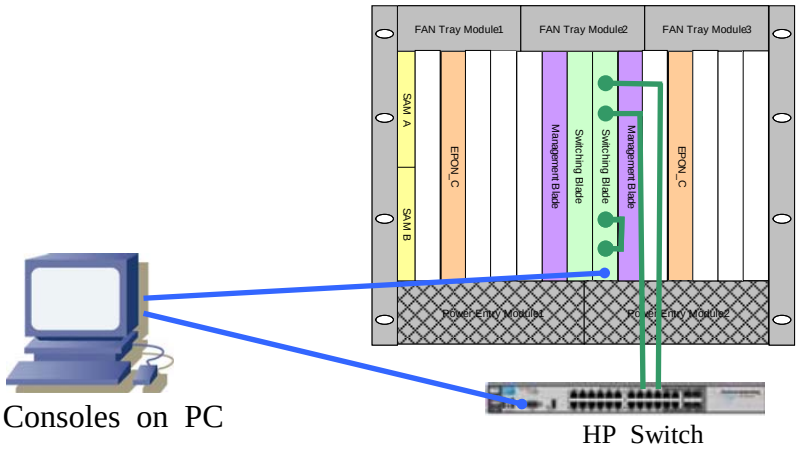
已完成其功能之驗證與測試，並進行 failover 之 HA 性能測試，結果顯示了當一塊 management blade (active) fail 時，能順利切換到另一塊 management blade (standby)，由原 standby SNMP agent 接手成為 active，接續封包監視計數等管理工作之進行，並不會造成管理功能與服務之中斷，以及影響整個 EPON 系統功能之執行。

5.3 Switch Blade 建置與驗證

如前所述，Switch Blade 是整個系統的資料傳輸路徑切換中心，並提供了上行的連接介面。本計畫已完成二套 Switch Blade 之建置，透過 ATCA 背板來做 1:1 的防護機制，並建構 Switch Blade 軟體，支援 Spanning Tree Protocol (STP)協定，以提供資料交換之功能。

Spanning Tree Protocol (STP)是用於交換器和交換器或橋接器之間互連時，避免其間的多重路徑形成迴路的協定。STP 已由 IEEE 802.1D 定義標準規範，以互相交換 BPDU 的訊息來偵測是否有迴路形成，並移除之間的迴路。STP 藉由強制使特定的多餘資料路徑進入備用的封鎖狀態，確保兩個網路設備之間在同一時間只會有一條路徑可以選擇。當網路之間的原先路徑斷線或有異動時，STP 演算法會重新組態 Spanning Tree 的架構以維持暢通的樹狀結構。

本計畫對支援 STP 之 Switch Blade，完成了功能測試與驗證，測試架構如圖五十所示，結果顯示 Switch Blade 之效能良好，能迅速適當的重組各 switch port 的角色並予封鎖或啟動，有效避免了多重路徑迴路之形成，將資料交換傳輸路徑維持於一暢通的樹狀結構中。另外，也進行了 failover 測試，顯示故障切換發生時，並不會造成 data plane forwarding 的中斷而影響了資料傳輸之進行。



圖五十、Switch Blade 功能測試架構與環境

5.4 CGOA EPON 平台整合與測試環境建置

為了發展、建構一符合電信等級開放平台架構(CGOA)之完整的 EPON 平台系統，本計畫將前述的 EPON Blade、Management Blade、Switch Blade 等各個模組單元，整合建置於一 ATCA Chassis 中，以提供一個開放的 EPON 平台。整個完整的 EPON OLT ATCA-based Chassis 系統業已整合建置完成，其系統實體如圖五十一所示。

EPON Blade

Management Blade

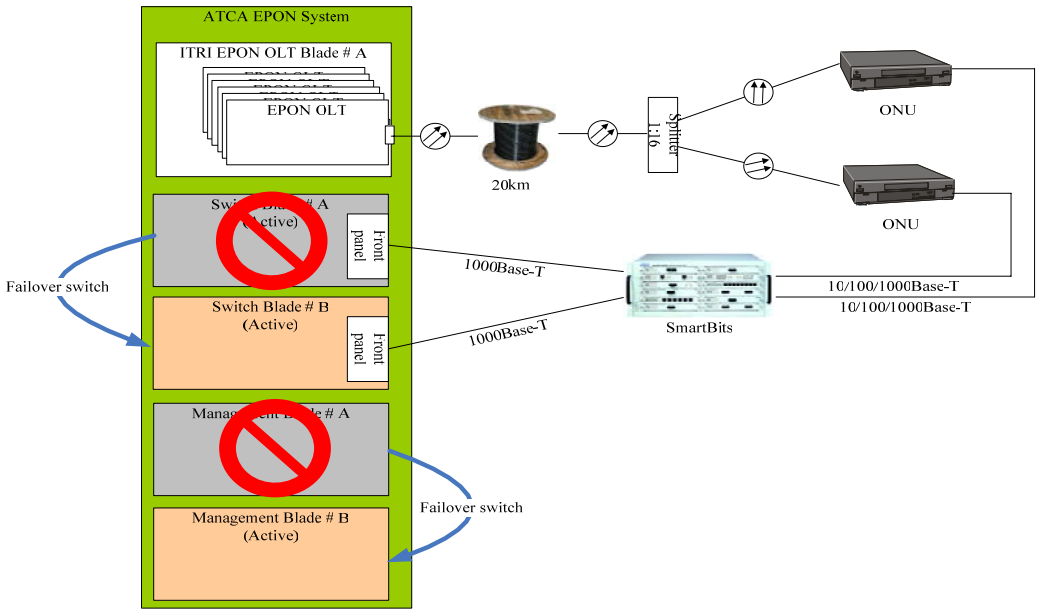
Switch Blade

EPON OLT
ATCA-based Chassis System

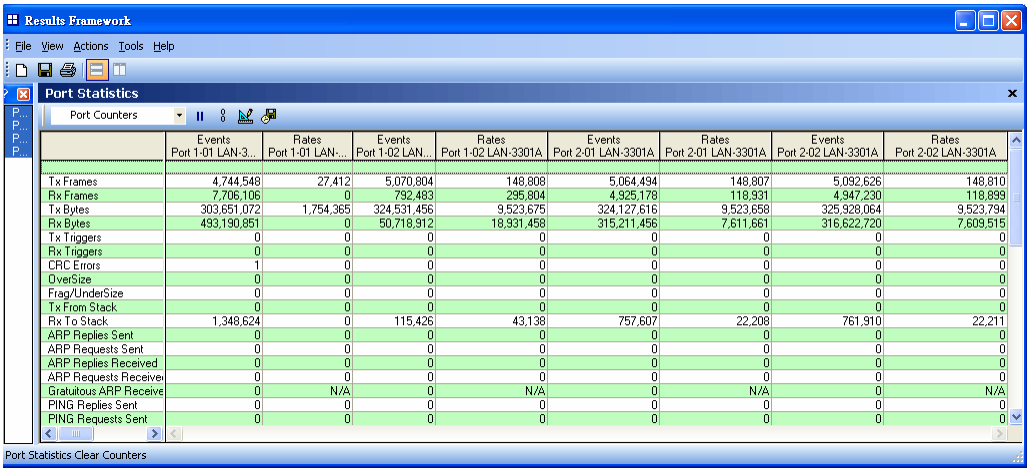
圖五十一、ATCA-based EPON Platform系統實體圖

有關整個 ATCA-based EPON OLT Chassis System 的整合測試環境，也建立完成，用以驗證 EPON 系統之功能與 HA 可靠度性能。整個系統的整合測試架構與環境，如圖五十二所示，以 Switch blade 透過 ATCA 內部 backplane 與 EPON OLT blade 相連，再將 EPON balde 的 EPON port 經由 20KM 光纖纜線及 splitter 連接至 2 台 ONU，並利用 SmartBit Traffic Analyzer 來傳送測試訊務資料(frame)，以驗證整個 EPON 網路之功能。更進一步，針對系統的故障防護能力與高可用度性能加以測試，分別檢測 Management Blade 與 Switch Blade 拔除或發生故障時，整個 EPON 系統之功能與運作是否會受到中斷或影響。如圖五十三所示的測試結果，顯示了 EPON 網路上訊務資料的傳送與接收都依然正常的運作，驗證本 EPON 具有良好之 HA 防護能力。

另外，本計畫也針對實務應用，架構了一套實務應用示範的測試系統環境，如圖五十四所示，設置一 Vedio Server，透過 EPON 網路，提供高畫質的視訊服務給終端的 HDTV 使用者，用以檢視及測試在實際的應用中，是否能滿足高品質視訊服務之需求。透過此測試架構，可進一步連接 Internet 網路，進行各種應用服務之測試與驗證，以提供後續的各項應用服務的試運轉之用，並可驗證本 EPON 網路，能提供一可靠的寬頻網路服務。

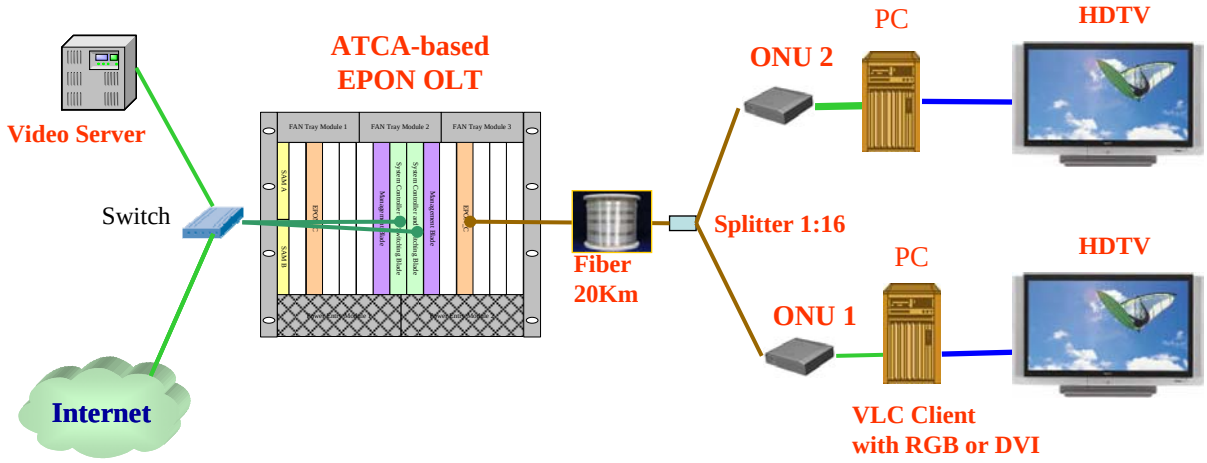


圖五十二、EPON平台系統整合測試架構



	Events Port 1-01 LAN-3...	Rates Port 1-01 LAN-...	Events Port 1-02 LAN...	Rates Port 1-02 LAN-3301A	Events Port 2-01 LAN-3301A	Rates Port 2-01 LAN-3301A	Events Port 2-02 LAN-3301A	Rates Port 2-02 LAN-3301A
Tx Frames	4,744,548	27,412	5,070,804	148,808	5,064,494	148,807	5,092,626	148,810
Rx Frames	7,706,106	0	792,483	295,804	4,925,178	118,931	4,947,230	118,899
Tx Bytes	303,651,072	1,754,365	324,531,456	9,523,675	324,127,616	9,523,658	325,928,064	9,523,794
Rx Bytes	493,190,851	0	50,718,912	18,931,458	315,211,456	7,611,661	316,622,720	7,609,515
Tx Triggers	0	0	0	0	0	0	0	0
Rx Triggers	0	0	0	0	0	0	0	0
CRC Errors	1	0	0	0	0	0	0	0
OverSize	0	0	0	0	0	0	0	0
Frag/UnderSize	0	0	0	0	0	0	0	0
Tx From Stack	0	0	0	0	0	0	0	0
Rx To Stack	1,348,624	0	115,426	43,138	757,607	22,208	761,910	22,211
ARP Replies Sent	0	0	0	0	0	0	0	0
ARP Requests Sent	0	0	0	0	0	0	0	0
ARP Replies Received	0	0	0	0	0	0	0	0
ARP Requests Receive	0	0	0	0	0	0	0	0
Gratuitous ARP Receive	0	N/A	0	N/A	0	N/A	0	N/A
PING Replies Sent	0	0	0	0	0	0	0	0
PING Requests Sent	0	0	0	0	0	0	0	0

圖五十三、整合EPON平台系統之訊務測試結果



圖五十四、Carrier Grade EPON平台的實務應用示範與測試之架構與環境

第6章 成果自評

6.1 本計畫產出物

第一年預計產出成果：

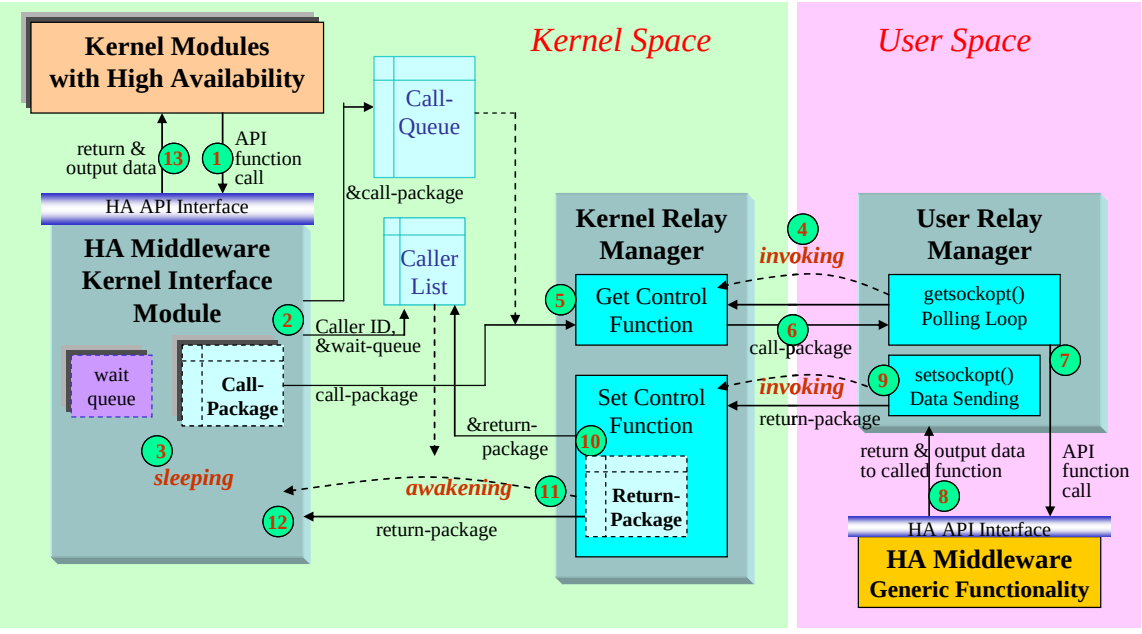
- 提出HA middleware實做方式的架構設計
- 針對現有Middleware缺失所提出的新架構設計與效能分析
  - ✓ 論文投稿與專利申請
  - ✓ 貢獻至相關 open source projects
  - ✓ 與業界或研究單位共同向 SA Forum提出標準
- 完成EPON系統可靠度測試環境建置與相關測試流程規劃

第一年執行成果：

- 提出HA middleware實做方式的架構設計
  - ✓ 完成Novel K-U bridge schemes for HA middleware設計

基於 HA middleware 的 Generic Functionality 之基礎，設計完成一特別的 Kernel-User Bridge 仲介轉換方法與機制之設計，將 kernel space 應用軟體的 HA APIs 需求，轉送至原本只支援 user space 的 Generic Functionality，再予以執行，以便對於 Linux 環境下 kernel space 及 user space 的兩種 HA 應用都能夠予以支援。

整個 Kernel-User Bridge 方法之設計如圖五十五所示，包含了 kernel space 的 Kernel-side Relay 與 user space 的 User-side Relay 二大機制。Kernel-side Relay 是支援 kernel space 中各 kernel module 程式的 HA–Middleware API Interface 介面呼叫之需求，負責將 kernel module 所呼叫的 API function 仲介轉送給位於 user space 的 User-side Relay，委由 User-side Relay 以 Application 的角色去向 HA Middleware 的 Generic Functionality 呼叫該 API function 之服務。在技術特性上，該設計採取了 generic oriented 一般性導向之設計，Kernel Relay Manager 與 User Relay Manager 等轉換機制為一般性的，與應用系統無相關性(application independency)，也可適用於其他軟體 API 的轉換支援。另外，也研擬了有效率的 polling 技術與 queuing 機制，讓 kernel space 與 user space 間的資料傳輸效率能達最佳化，並在系統資源、CPU 之額外消耗與負擔上，取得良好之平衡。



圖五十五、Novel K-U bridge 機制之設計

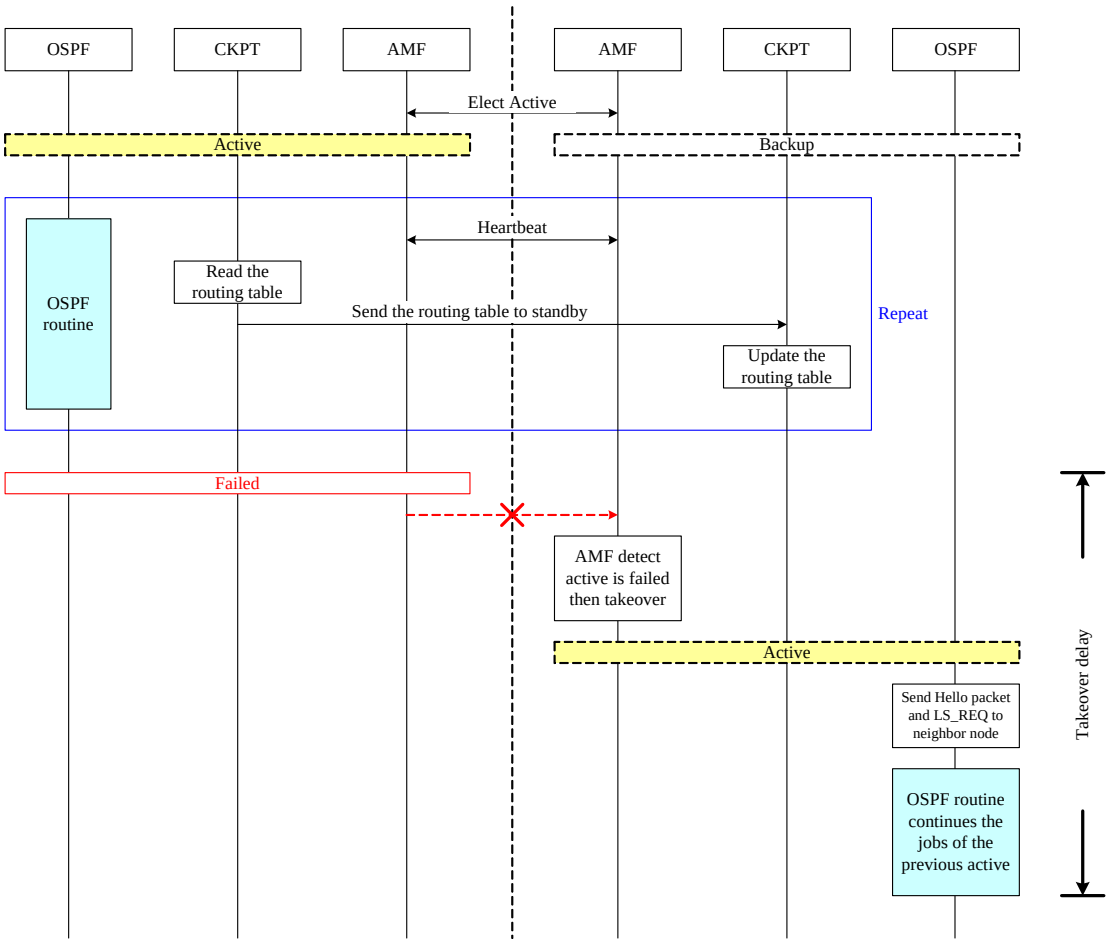
所完成的 K-U bridge schemes，能有效支援 HA middleware 在 Linux 環境下 kernel mode 與 user mode 的兩種應用模式，並驗證其效能，結果在效能及效率上有相當良好之表現，將可符合 carried grade 應用之要求。

✓ 預計明年年初申請專利，且提出標準draft至SA Forum標準會議

➢ 針對現有middleware缺失所提出的新架構設計與效能分析

✓ 提出Fast recovery mechanisms of routing paths for HA middleware之方法

在一般的模式下，當 OSPF Router 中斷服務之後，若 Router 重新加入網路之後需要至少 20 秒之後才能夠重新加入網路，繼續轉送封包的工作，此方法對於企業等級的網路服務的中斷時間要求是不夠的，因此，本研究成果利用 HA middleware Generic Functionality 所提供的 AMF 以及 checkpoint 服務來達到 High Availability 的目的。流程圖如圖五十六所示，在一開始 Router 組會彼此決定溝通選擇出 Active，其他的 Router 皆為 Standby，在同一時間只會有一台 Router 負責轉送封包至正確的網路，且對於 Neighbor Router 而言，會視此 Router 組為一個轉送的節點。



圖五十六、Fast recovery mechanisms of routing paths for HA middleware 的流程圖

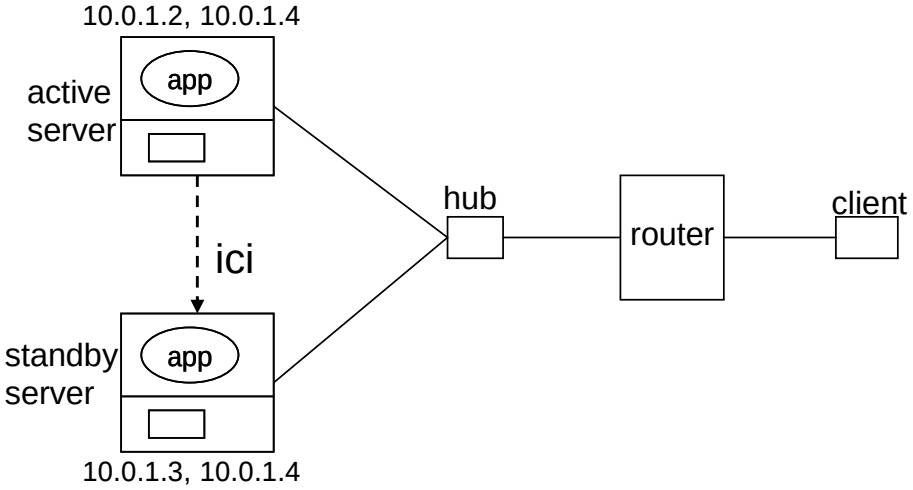
執行期間，Active 與 Standby 中的 AMF Service 會定期送出 Heartbeat，主要目的是監測 Active 是否仍在運作，若否，則 Standby 將會立刻接手 Active 所執行的工作。除此之外，Active 會從 Kernel Space 取出 Routing Table，利用 checkpoint 方法將 Routing Table 傳送給 Standby，當 Active 中斷服務之後，Standby 可以立刻接手工作，且 Standby 接手工作之後，會立刻送出 Hello 以及 LS_REQ 封包，立刻向 Neighbor Router 做註冊以及詢問 Neighbor Router 的路由資訊，而不等 timeout，這樣可加快接手的速度。

如圖五十七所示，整個換手的延遲時間是從 Active fail 之後開始計算，當 standby 沒有接收到 Active 的 Heartbeat，便開始換手工作，直到 Standby 可以開始轉送封包為止，根據圖五十七所示，延遲時間可分為幾個部分，本成果是加快其中的 Hello 以及 LS_REQ 封包送出的速度，用以加快接手延遲。如表二所示，利用本研究成果所設計出來的備援方式所得到的 Recovery Delay 很明顯的小於一般的方式，因此利用 High Availability 所設計出來的 Router 在企業等級網路內路由節點的維護，可以扮演相當重要的角色。

表二、有無備援機制下的 OSPF 路由服務恢復時間

Test No.	無備援機制	備援路由資訊	OSPF Redundancy
1	20.7	10.41	0.417
2	22.32	9.95	0.419
3	21.77	11.51	0.420
4	20.55	11.07	0.421
5	21.66	10.27	0.422
單位：ms			

✓ 提出TCP takeover designs in HA middleware之方法



圖五十七、TCP takeover designs in HA middleware 的實驗測試環境

本研究成果設計出一個如圖五十七所示的實驗環境，用以進行 TCPCP 與 HA middleware 的相關研究。我們預設的情況是 active server 原本與 client 擁

有正常的連線，然後假定有一突發的狀況發生使得 active server 運作失常。此時，在同一網域的 standby server 必須盡速的取代原本的 active server 並且繼續與 client 連線而不被 client 察覺。

在 server 端的應用程式方面，我們選定的是 telnet server。我們採用了一個簡單的 telnet server 程式，並且修改其程式碼使其利用 tcpcp 所提供的連線轉移功能與 HA middleware Generic Functionality 所提供的 checkpoint service。

程式的主要基本概念為利用所提供的 checkpoint service，持續的將 TCPCP 所擷取出的 ICI 傳送至 standby server。更正確的說，由於此服務為交談式服務，所以 TCP 連線狀態只有 client 傳送 command 給 server 時才改變。我們是在 client 下命令以及 server 回傳結果給 client 之後去擷取 ICI 並更新至 standby。如此可節省 active 與 standby 之間大量的 ICI 交換頻率。

當 active server 故障時，HA middleware Generic Functionality 所提供的 AMF 會偵測到此錯誤並且將 standby server 的身分轉變為 active。Standby server 此時則利用其所接受到的最新的 ICI 將 TCP 連線恢復。在實作過程中，必須要注意的是每當 TCPCP 要取得 ICI 時，會將該連線的 socket 隔離(並沒有 close，但卻也無法再使用)，所以我們必須先將 socket close，然後另外開一個新 socket 來將 ICI 設回，這樣做才能連續的存取與設回 ICI。

- ✓ 正在申請2件專利
- ✓ 預計年底投稿國際研討會論文2篇 (International Service Availability Symposium 或International Conference on Dependable Systems and Networks)

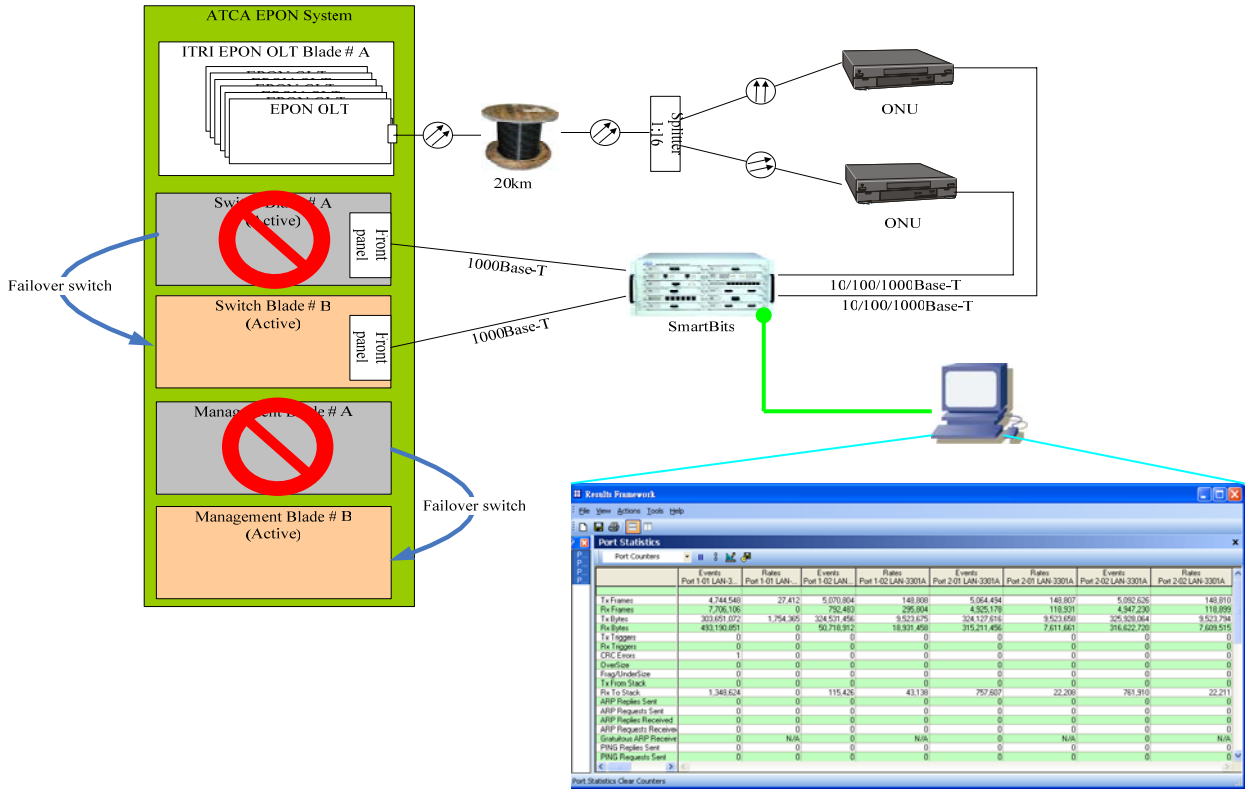
➢ 完成EPON系統可靠度測試環境建置與相關測試流程規劃

- ✓ 完成CGOA EPON Platform整合 (如圖四十四所示，但不包含第二年的兩套EPON卡板及四套Processor卡板)
  - ATCA shelf x 1
  - EPON Blade x 2
  - Management Blade x 2
  - Switch Blade x 2
- ✓ 完成CGOA EPON Platform單元測試
  - EPON Blade 測試：
    利用 SmartBit Traffic Analyzer 流量分析儀來傳送測試訊務(frame)，以檢視及驗證 EPON Blade 網路 Downstream 及 Upstream 兩方向之資料封包傳輸功能，結果顯示資料的傳送與接收功能都能正常的運作，且 throughput 及 latency 等性能皆能符合規格需求。另外，針對封包流失(packet loss)的情形加以量測，結果顯示並無任何封包遺失。
  - Management Blade 測試：
    針對兩套相互備援的 Management Blade 及其上所建置的 HA Middleware 與管理軟體，進行 failover 之 HA 性能測試，結果顯示了當一塊 management blade (active) fail 時，能順利切換到另一塊 management blade (standby)，由原 standby SNMP agent 接手成為 active，接續封包監視計數等管理工作之進行，並不會造成管理功能與服務之中斷。
  - Switch Blade 測試：
    對支援 STP(Spanning Tree Protocol)之 Switch Blade，完成了功能測試與驗證，結果顯示 Switch Blade 之效能良好，能迅速適當的重組各 switch port

的角色並予封鎖或啟動，有效避免了多重路徑迴路之形成。另外，也進行了 failover 測試，顯示故障切換發生時，並不會造成 data plane forwarding 的中斷而影響了資料傳輸之進行。

– CGOA EPON 平台測試：

有關整個 ATCA-based EPON OLT Chassis System 的整合測試環境，也建立完成，用以驗證 EPON 系統之功能與 HA 可靠度性能。整個系統的整合測試架構與環境，如圖五十八所示，以 Switch blade 透過 ATCA 內部 backplane 與 EPON OLT blade 相連，再將 EPON balde 的 EPON port 經由 20KM 光纖纜線及 splitter 連接至 2 台 ONU，並利用 SmartBit Traffic Analyzer 來傳送測試訊務資料(frame)，以驗證整個 EPON 網路之功能。更進一步，針對系統的故障防護能力與高可用度性能加以測試，分別檢測 Management Blade 與 Switch Blade 拔除或發生故障時，整個 EPON 系統之功能與運作是否會受到中斷或影響。依據測試結果，顯示了 EPON 網路上訊務資料的傳送與接收都依然正常的運作，驗證本 EPON 系統具有良好之 HA 防護能力。



圖五十八、ATCA-based EPON OLT Chassis System的整合測試環境

6.2 預期效益

國際電信服務業者在佈建新的基礎建設及服務設備，如：EPON、3G、WiMax、IP Multimedia System(IMS)等，將會趨向採用電信等級開放平台(Carrier Grade Open Platform)。

藉由遵循PICMG (PCI Industrial Computer Manufacturers Group) 3.x 規範的硬體平台標準，以及Service Availability Forum (SAF)定義的高可靠度中介軟體(High Availability Middleware)介面規格，將可以大量採用商業化(COTS—Commercially Off-The-Shelf)元件(Building Blocks)，縮短產品開發的時程、成本與風險。依據In-Stat在2005年2月的市場預估，2009年開放平台市場規模可達120億美金，且2004-2009年間的年成長率可達104.9%。此一開放平台潮流也將是國內產業升級的契機。基於國內過去在電信網通相關產品所建立的品牌與攻佔市場的能力，必須在核心技術上要有突破性的提升，更要有主導技術的基本概念。面對著這一波新的市場趨勢與新需求，要能夠先一步地掌握技術發展，如此才能做出真正有差異化的技術與產品。因此，本計畫所結合產學研各界之能量完成的寬頻電信等級開放平台核心技術，將可創造台灣電信產業的藍海，並對技術研發紮根的重要基礎。預期對台灣整體電信產業的影響效益，條述如下：

- 國內寬頻設備產業主要專長在於硬體與系統生產製造能力，本計畫將可建立國內產業目前所欠缺的電信等級通訊軟體 Reliability 核心技術，並深入掌握其中影響效能之關鍵技術，而有效提升國內寬頻設備的可靠度。另外，也將使國內產業由電信等級開放平台產業鏈的元件(Building Blocks)切入，並進一步促成更多產學研的合作機會，建立更多電信等級應用服務的完整平台，甚至是系統解決方案，逐步在價值鏈中向上提升。

- 結合產學研力量，建立高可靠度局端設備研發能量，有助於我國廠商開發電信等級之局端通訊設備，取得FTTx及高階電信設備市場商機。本計畫開發出全球第一套ATCA-based EPON系統，是採用Open Communication Platform的設計理念開發完成的，此一開放式標準平台的設計趨勢已普遍為全球國際大廠所接受，主要的優點不但可以縮短產品的開發成本及時程外，由於各軟硬體介面均已標準化，很多技術及Building blocks都可以reuse或買得到，這尤其對於國內以中小企業為主的廠商而言特別重要，國內廠商不再需要投入大量人力物力，從無到有地建立自有的平台，廠商可依各自的優勢，更加專注投入高附加價值功能的開發，增加市場的區隔性。

- 掌握開放通訊平台技術趨勢，協助我國廠商切入高階電信等級通訊設備之模組與系統市場，由產製 Ethernet Switch，SOHO Router，xDSL CPE 等中低階通訊產品轉型為高階 Edge/Access 端產品提昇；另外，也促成資訊產業進入電信產業之競爭力，提升我國ICT (Information and Communication Technology)產品之技術層級與附加價值並協助ICT產業升級：

  - ✓ 通訊產品: OEM/ODM周邊產品-->具品牌之Edge端產品
  - ✓ 資訊產品: 硬體生產製造-->硬體附加軟體應用價值
  - ✓ 推動架構於新興寬頻網路之設備與配置

- 建置EPON、ATCA、HA middleware之互通與測試環境，提供廠商驗證產品研發的標準符合性與互通性，短期立即可促使我國電信等級通信設備垂直分工的產業鏈形成，而國內廠商將可持續投入開放硬體平台相關組件的研發製造，如風扇、電源供應器、記憶體、機箱、卡板及背板等。中期藉由本計畫所建立的HA middleware高可靠度技術及ATCA系統平台，將可我國電信設備產業由可用度99.9%、99.99%逐步推升至電信等級99.999%需求，而朝向開發能行銷全球的電信系統產品，如電信等級網路設備或伺服器等，以提高競爭門檻與附加價值；而長程效益，本計畫

所完成ATCA-EPON系統在中華電信試運轉將使其他電信服務營運商追隨加入並開始採用高可靠度電信等級開放平台上提供電信等級服務，如含語音、視訊及資料流的Triple play服務，以此「寬頻電信等級開放平台」為基石帶動我國電信應用服務軟體的發展，並使我國電信業在國際產業鏈的角色由製造業轉型為高附加價值的服務業，以推動我國電信產業朝向另一波高峰。

參考文獻：

[1] Bar, Moshe. OpenMosix, Proceedings of the 10the International Linux System Technology Conference (Linux-Congress 2003), pp. 94102, Oct. 2003.

[2] Bryan Kuntz and Karthik Rajan. MIGSOCK Migratable TCP Socket in Linux. Master's Thesis, Carnegie Mellon University, Feb. 2002.

[3] http://tcpcp.sourceforge.net/

[4] http://tcpcp2.sourceforge.net/

[5] W. Almesberger. TCP Connection Passing. In Proceedings of Ottawa Linux Symposium 2004, vol. 1, pp. 9–21, July 2004.

[6] T. Ikebe, Y. Kawarakaki, and J. Yamanaka. Practical TCP Session Take-over Method for High-availability Network Service. In Proceedings of 6th Asia-Pacific Symposium on Information and Telecommunication Technologies, 2005 (APSITT 2005), pp. 1–6, Nov. 2005.

[7] S. Brossier, F. Herrmann, and E. Shokri. On the Use of the SA Forum Checkpoint and AMF Services. In Proceedings of International Service Availability Symposium 2004 (ISAS 2004), May 2004, Munich, Germany.

[8] Fábio Olivé Leite. Load-Balancing HA clusters with No Single Point of Failure. In Proceedings of the 9th International Linux System Technology Conference (Linux-Kongress 2002), pp. 122–131, Sep. 2002. http://www.linux-kongress.org /2002/ papers/lk2002-leite.html

[9] J. Mahdavi and S. Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Technical note sent to the end2end-interest mailing list, January 8, 1997.

[10] Y. Chen, C. Chen and C. Huang. Experience in Developing a High Availability and Continuous TCP Using OpenAIS and TCPCP. Submitted to International Service Availability Symposium 2007 (ISAS 2007) (本計畫投稿論文)

附件二

可供推廣之研發成果資料表

■ 可申請專利　□ 可技術移轉　　日期：__年__月__日

國科會補助計畫	計畫名稱：寬頻電信等級開放平台建置計畫 計畫主持人：張仲儒 計畫編號：　　學門領域：電信國家型計畫
技術/創作名稱	支援高可靠網路協定服務的 High Availability Middleware
發明人/創作人	簡榮宏、陳健、蔡嘉泰等
技術說明	中文： 為了提供電信等級的網路系統及服務，擴充了 High Availability Middleware 的增強功能，以支援 Layer3/Layer4 網路協定及其他網路應用能實現高可靠的服務。另外，也對執行於 Linux-based Kernel Mode 的協定服務，提供了使用的支援。針對 OSPF (Open Shortest Path First)路由協定以及 TCP 協定，基於 Redundancy 的備援架構，協同運用了 High Availability Middleware 之 Generic Functionality 中的可用度管理與資料備份等機制，同步備份了重要的協定資訊給 Standby 端，如 OSPF 的 Routing Table 及 TCP 的 Connection 等資訊，並都加入了快速復原的設計考量，以建立他們運作備援及故障切換的能力。
	英文： Expanding the High Availability Middleware with Enhanced Functionality can support the high available services of Layer3/Layer4 protocols and kernel mode applications. This design uses the Generic Functionality of High Availability Middleware to backup the protocol information such as Routing Table in OSPF and Connection Information in TCP, and to achieve the high availability.
可利用之產業及可開發之產品	• Telecom, Communication, Networking, and Service Provider • Carrier grade Router and network system • HA-Compliant Internet servive system
技術特點	1. 於 High Availability Middleware 中增加了對 Layer3/Layer4 網路協定及其他網路應用的支援。 2. 透過 Kernel-User bridge 的機制，提供 Linux-based Kernel Mode 應用服務的使用支援。
推廣及運用的價值	• 協助業者可快速投入及進行電信等級設備與服務系統之開發，以滿足使用者的高可靠度及不中斷服務之需求。 • 加速及促進電信等級服務的基礎建設及設備的佈建。 • 提昇國內寬頻產業的核心技術競爭能力及產品之技術層級與附加價值，並提供產業技術升級及切入高階電信市場的契機。

※　1. 每項研發成果請填寫一式二份，一份隨成果報告送繳本會，一份送　貴單位研發成果推廣單位（如技術移轉中心）。

※　**2. 本項研發成果若尚未申請專利，請勿揭露可申請專利之主要內容。**

※　3. 本表若不敷使用，請自行影印使用。

Experience in Developing a High Availability and Continuous TCP Using OpenAIS and TCPCP

Ying-Yu Chen[1], Chien Chen[2] and Chia-Yuan Huang[1]

[1] Information & Communications Research Laboratories,
Industrial Technology Research Institute,
195 Chung Hsing Rd., Sec. 4, Chu Tung, Hsinchu 310, Taiwan
{itri404393, ricehuang}@itri.org.tw
[2] Department of Computer Science, National Chiao Tung University,
Hsinchu, Taiwan, R.O.C
cchen@cis.nctu.edu.tw

Abstract. It has become one of the basic requirements for the service providers to deliver highly available services to meet the customers' critical needs. However, a highly available service does not guarantee that the service is delivered continuously from the user's point of view. In this paper, we share the experience of developing high-availability and continuous TCP using open source OpenAIS and TCPCP/TCPCP2. We describe the main system design for building such systems. We specifically discuss two of the problems that need to be overcome under this kind of design model. For each problem and its corresponding resolution, we show a simulation using ns2 simulator, which provides a deeper insight into the problems for further studies. We also develop a simple application which uses OpenAIS and TCPCP2 to achieve its high availability and service continuity. The results indicate that a high availability and service continuity service can be obtained with minor degradation in performance.

Keywords: High availability, service continuity, TCP, OpenAIS, TCPCP (TCP connection passing).

1 Introduction

As technologies advances, users are getting more and more dependent on network services in their daily lives. Therefore, it is very important that the services delivered to the users are highly reliable and meet the customers' expectation. To achieve high availability, a system usually contains redundant components so that when the main server fails, it can be replaced immediately by another. However, a service with high-availability, e.g., 99.999% uptime, does not guarantee that the service is delivered continuously from the users' point of view. In case where the service has to be transferred from one host to another, e.g. failover and switchover, maintaining service continuity means that the service is transferred with minimal degradation in performance, and without users being aware of it. The properties of possessing high availability and service continuity are referred to as service availability [1].

In this paper, we focus on the failover of TCP connections from one host to another since lots of important network applications are built on top of TCP such as HTTP, FTP, SMTP, SSH, etc. It is hard to make these applications continuous unless the underlying TCP can be made continuous in case of failover. The major challenge is that TCP is a connection-oriented protocol and many of the connection parameters such as sequence number, TCP flags, send/receive buffers, etc., which are kept in the kernel of the active node, have to be synchronized with other standby nodes.

In this paper, we share the experience of developing high availability and continuous TCP using the open source OpenAIS [2] and TCPCP/TCPCP2 [3][4]. We describe how OpenAIS and TCPCP/TCPCP2 can be used to develop services with improved service availability. Then, we elaborate on two problems that occur under this design model. In the case where TCP data flows from the client to the server and a failure occurs within the interval between two checkpoints, one problem arises when the standby server tries to resume the service using the last checkpoint information. Since data that have already been acknowledged have been deleted from the client's TCP send buffer, it is not possible for the client to retransmit those data to the server. One solution is to synchronize TCP acknowledges from servers with checkpoint interval. That is, the acks will only be sent from server every checkpoint interval. However, delaying sending acks from server side would have impact on the TCP throughput. In this paper, the numerical results demonstrate that acks should not be delayed more than 10~40ms depending on the link rates.

The other is about what TCP congestion control status should be set to the standby server when the active server fails, since after all it is the *last* checkpoint information that we use to resume the TCP connections. However, the congestion window size recorded in the last checkpoint may no longer reflect the current network condition. Therefore, we develop a window size prediction method to observe how the performance is affected when a simple mechanism is added to support service availability. The simulation results are shown using ns2 [5] simulator. We also reports the results of a simple application that we developed to observe how the performance is affected when extra overhead is added to support service availability.

The rest of the paper is organized as follows. Section 2 describes how TCP connection passing tools TCPCP/TCPCP2 can be used with OpenAIS to develop service availability services. In Section 3, we describe the problems mentioned above in detail and the respective solutions. The simulation results are shown using ns2 simulator. In Section 4, we show the performance of our simple service availability application. The paper concludes in Section 5.

2 Background and Design Model

2.1 OpenAIS and TCPCP/TCPCP2

OpenAIS is the Linux middleware implementation of the SA Forum's Application Interface Specification (AIS) [6]. SA Forum AIS standardizes the interfaces between SA Forum compliant high availability middleware and service applications so that service availability application software can be developed independent of the

underlying platform. Authors in [7] describe the use of Application Management Framework (AMF) and Checkpoint Service in SA Forum AIS to implement high availability services, where AMF [8] supervises redundant resources within the server cluster to deliver the service with no single point of failure and Checkpoint Service [9] is used to record checkpoint data, which can be retrieved to resume the service after the failure.

TCPCP is the implementation of the mechanism that provides APIs for applications to pass the ownership of TCP connection endpoints from one host to another. TCPCP2 is another form of TCPCP, and is created based on TCPCP. The two are similar in principle and both require the kernel modification in the servers. However, the major advantage over other TCP connection passing methods such as MIGSOCK [10] is that the client side does not have to be modified. Since TCPCP and TCPCP2 are quite similar in nature, for the rest of the paper, the description is based on TCPCP2.

TCPCP2 provides a set of APIs to be used by the application to allow the TCP socket information (SI) (including source/destination IPs and ports, TCP flags, sequence number, send/receive buffer, etc.) to be retrieved/set back from/to the kernel. Therefore, the service process on host A can use TCPCP2 to retrieve SI, and by any means, which is mentioned in Section 2.2, send the retrieved SI to host B. Host B can then set the received SI back to its kernel. Sample procedures to take over the TCP connections can be found in [11]. The APIs provided by TCPCP2 includes:

- **tcpcp_stop( ).** Stop the connected TCP socket form sending and receiving packets.
- **tcpcp_get( ).** Retrieve the TCP socket information.
- **tcpcp_set_si( ).** Set the SI into TCP socket.
- **tcpcp_start( ).** Allow the TCP socket to resume sending and receiving packets.

SA Forum AIS and TCP session takeover are currently included in the OSDL Carrier Grade Linux Requirements and Roadmap [12], respectively.

2.2 Achieving Service Availability

Neither TCPCP nor TCPCP2 describe when to pass the TCP connections or how to transfer the retrieved SI to other hosts. Our approach is to combine the ability of TCPCP2, and the AMF and Checkpoint Service in OpenAIS to achieve our goal. An overview of the design model is shown in Fig. 1.

During normal operation, the application status and TCP SI (retrieved by tcpcp_get( )) of the active server are periodically transferred to the standby nodes in the cluster using the Checkpoint service. When the active fails, the error is detected by the AMF, which then dictate one of the standby nodes to become active. The newly active node can now use the last checkpoint data received from the failed active node to restore the application and TCP status so that it can continue to serve its connected clients. Notice that the ways of redirecting the IP packets which were originally intended for the failed node to the standby node are beyond the scope of this paper. Some references can be found in [13] and [14].

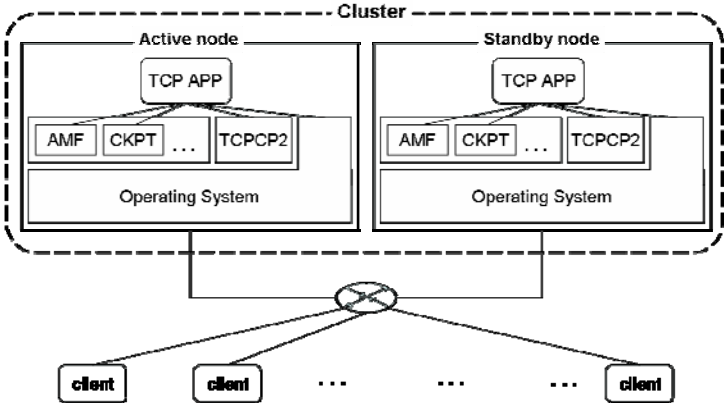


Fig. 1. Design model of a service availability application.

3 Problems to Overcome

Despite the simple system design described above, TCP itself is a complicated protocol and was originally not designed for such usage. Projects of TCPCP and TCPCP2 are still advancing and some problems still leave to be solved [13][15]. In this section, we describe two of them and their corresponding solution, and show the simulation results using ns2 simulator.

3.1 Delaying Acks on Server Side

Consider the situation that the data only flows from the client to the server. Assume that the fail occurs between the last checkpoint and the next checkpoint, as shown in Fig. 2. During the last checkpoint and the fail, the client's TCP may have sent some data with sequence number 4, 5 and 6 out of its send buffer to the server, and the server's TCP may have acknowledged the data in its receive buffer. Therefore, the client will clean the data with sequence number 4, 5 and 6 in its send buffer. The problem arises when the standby server tries to resume the service using the last checkpoint data with sequence number 1, 2 and 3 during a failure, since the client's TCP send buffer contains no data with sequence number 4, 5 and 6. The client is unable to retransmit the data with sequence number 4, 5 and 6 to the server.

A possible solution is to delay the server from sending acks to the client until the server application has retrieved the received data and reached the next checkpoint. That is, the acks will only be sent from server every checkpoint interval. By this it is ensured that even the server fails between checkpoints and the standby server is resumed to the last status, the client's TCP send buffer still contains the data that need to be retransmitted. The drawback of such solution is that the estimation of round trip

time (RTT) at the client would be affected, which may lead to a TCP throughput degradation.

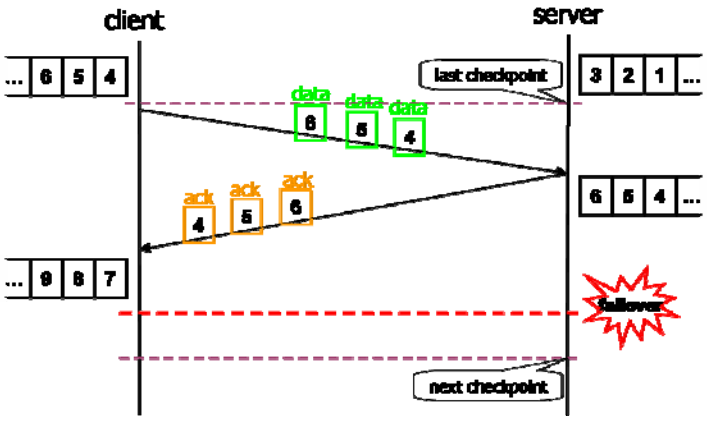
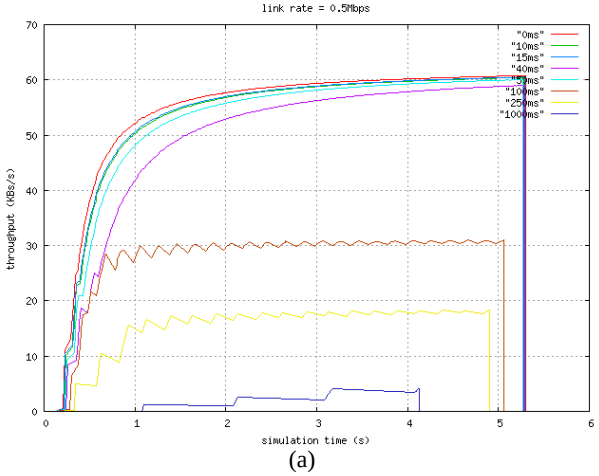


Fig. 2. Illustration for the checkpoint problem when server fails between checkpoints.

We simulate the delay acks by modifying ns2 simulator to support this mechanism. A client node and a server node are setup, and between them is a router that forwards the packets from the client to the server. The client keeps sending ftp data to the server. The throughput is measured under different delay intervals. The results are shown in Fig. 3(a)~3(c), where the link rates are set to 0.5, 1 and 2Mbs, respectively, and the legend on the upper-right corner in each figure is sorted in the sequence of the curves from top to bottom. It is interesting to observe that the throughput does not decrease sharply unless the delay interval is increased to a certain value, and such intense decrease is at least half the throughput of the best curve (i.e., no checkpoint). The results show that the delay acks mechanism degrades the TCP performance significantly when the acks are postponed more than 40ms, 15ms and 10ms for the link rates 0.5, 1 and 2Mbps respectively. Therefore, the checkpoint interval would have to be bounded within a very small range if delay acks solution is to be applied.



(a)

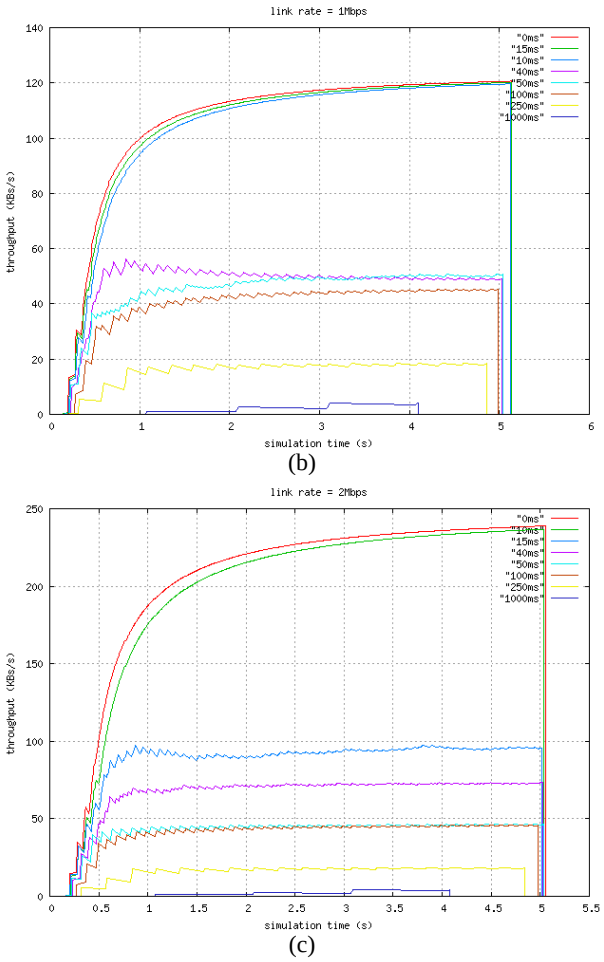


Fig. 3. The throughput when different checkpoint intervals are used. The network link rates are set to 0.5, 1 and 2 Mbps. (a) Link rate = 0.5 Mbps. (b) Link rate = 1 Mbps. (c) Link rate = 2 Mbps.

3.2. Congestion Window Prediction

If the failover should happen, the checkpoint data used to resume the service is, after all, not so *fresh*. More specifically, the congestion control data recorded in the SI may no longer reflect the current network condition. Consequently, an improvement may be considered by allowing different strategies to be applied to the congestion control status of the resumed TCP connections.

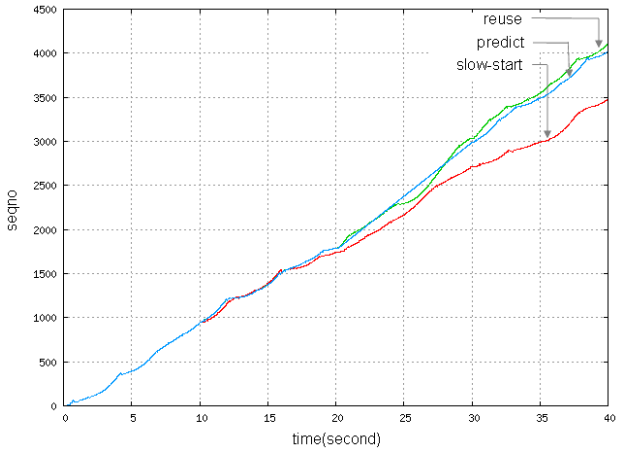
We conduct the simulation using ns2 and compare the performance when the following strategies are used to set the congestion window size of the resumed TCP connection.

- **slow-start.** Set the congestion window to one.
- **reuse.** Use the congestion window size recorded in the last checkpoint.
- **predict.** Set the congestion window to the well-predicted size.

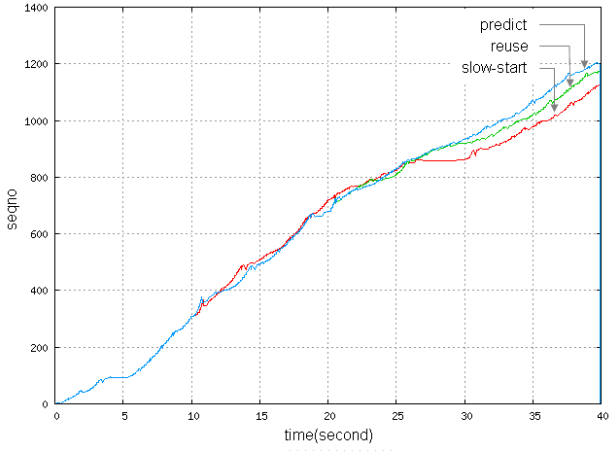
In this paper, we propose to use the predicted congestion window size by

$$\text{Congestion Window Size}_{\text{predicted}} = 1.22/(\text{RTT}*\text{sqrt}(\text{Loss})), \quad (1)$$

which can be simply derived from [16].



(a)



(b)

Fig. 4. The growth of sequence vs. time when different congestion window is set to the resumed TCP connection. (a) Delay box loss rate = 0~0.01. (b) Delay box loss rate = 0~0.09.

To simulate the network condition, we put a delay box between the server and the client. The link bandwidth is 2Mbps. The server sends ftp traffic to the client for 40 seconds. At the time 10, 20 and 30 secs, the traffic is interrupted and resumed again. The reuse strategy described above uses the congestion window size retrieved 0.5 sec before the interruption to resume the connection. Figure 4(a) and 4(b) show the

comparison results when the loss rate of the delay box is set to 0 ~ 0.01 and 0 ~ 0.09, respectively (in the simulation, a sequence number represents a sent packet). It can be observed that when the loss rate is low (Fig. 4(a)), the slow-start strategy obviously has the slowest growth of sequence number. On the contrary, when the loss rate is high (Fig. 4(b)), the gap between slow-start and the other two is reduced. It can also be observed, although not obvious, that the predict strategy seems to outperform the reuse strategy when the loss rate is high. This is because the higher loss rate also means that the network condition is more unstable. Therefore, a well-predicted congestion window may lead to better performance than reusing the old one.

4 A Simple Service Availability Application

It can be anticipated that tradeoff exits between the degree of service availability and the service performance. The more frequent checkpoint synchronization between active and standby nodes, the less processor time is spent on the original service.

We develop two versions of a simple file transfer program in order to observe the difference in performance, one with the service availability using OpenAIS and TCPCP2, and the other without. The program starts to send a file to the client once the connection is established, and terminates after the completion of the transfer.

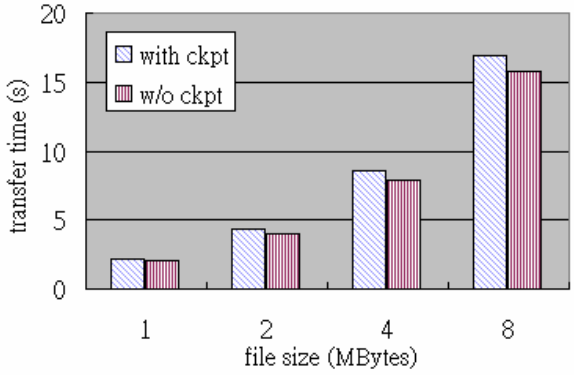


Fig. 5. Comparison of transfer time needed for different file sizes when the program is with and without service availability. The checkpoint interval is set to 0.5 sec.

Figure 5 compares the time needed to transfer files of different sizes to the client, with the checkpoint interval of the service availability version set to 0.5 sec. The result shows that on average, the service availability version takes 6.3% more time than the ordinary version to transfer a file. Figure 6 shows the time needed to transfer an 8Mbytes file to the client, with different checkpoint interval settings. It can be observed that the major decrease in transfer time is when the checkpoint interval is set from 0.1 to 0.2 sec, and the decrease attenuates as the checkpoint interval increases. Figure 7 depicts the growth of sequence number when there are no checkpointing, checkpoint interval = 0.5 sec and checkpoint interval = 0.3 sec, respectively. For the case with checkpoint interval = 0.5 sec (red cross curve), the growth of sequence number is suspended for a tiny moment at 0.5+ sec because the program is retrieving

and setting TCP SI. This can also be observed for the case with checkpoint interval = 0.3 sec at 0.6+ sec (green dot curve).

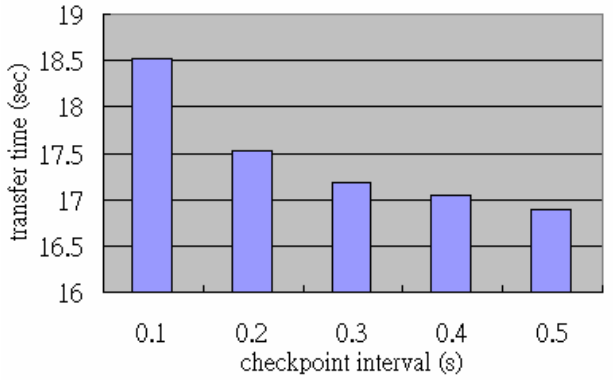


Fig. 6. The time needed to transfer an 8 MBytes file when checkpoint interval is 0.1 ~ 0.5 sec.

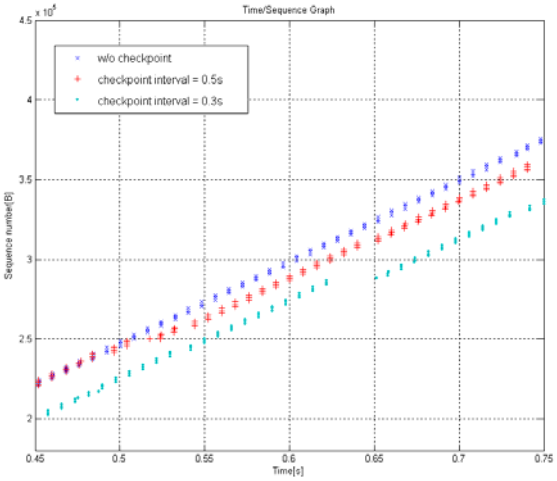


Fig. 7. The growth of sequence number during 0.45 ~ 0.75 sec.

5 Conclusion and Future Works

In this paper, we share our experience in developing high availability and continuous TCP using open source OpenAIS and TCPCP/TCPCP2. We show that by delaying sending acks from the server to the client until after the server application has retrieved the received data and finished checkpointing, the data in the client's send buffer can be prevented from being cleared, and thus making TCP recoverable using the last checkpoint. However, the delay can not be too long otherwise the throughput would be severely impaired. We also show that when TCP connections are resumed

from the checkpoint data, the congestion window could be well-predicted and set, to improve or preserve the original performance. Even though our simple congestion window prediction function shows minor performance improvement, a more complicated predication function will be invested in the future for further performance improvement. Finally, we display the performance result of a simple file transfer application that uses OpenAIS and TCPCP2 to achieve its service availability. The numerical results show that on average it uses 6.3% more times than the ordinary version to transfer a file when checkpoint is performed every 0.5 second.

The numerical results and the practical implementation presented in this paper provide only a partial inspection on the practicability of service availability TCP-based services. We will further our research on how to strengthen the weaknesses mentioned in [13] and [15], to facilitate the development of robust service availability TCP-based services.

Acknowledgments. We would like to thank Chih-Chiang Yang, Hsin-Fan Chen, Ching-Chun Kao and Lo-Chuan Hu for their support on information about ns2 and programming skills.

References

1. Service Availability Forum. Standards for a Service Availability Solution. http://www.saforum.org/about/solution_backgrounder.pdf
2. http://developer.osdl.org/dev/openais/
3. http://tcpcp.sourceforge.net/
4. http://tcpcp2.sourceforge.net/
5. http://www.isi.edu/nsnam/ns/
6. SA Forum Application Interface Specification AIS B.01.01
7. S. Brossier, F. Herrmann, and E. Shokri. On the Use of the SA Forum Checkpoint and AMF Services. In Proceedings of International Service Availability Symposium 2004 (ISAS 2004), May 2004.
8. SA Forum Application Interface Specification: Availability Management Framework SAI-AIS-AMF-B.01.01
9. SA Forum Application Interface Specification: Checkpoint Service SAI-AIS-CKPT-B.01.01
10. B. Kuntz and K. Rajan. MIGSOCK: Migratable TCP Socket in Linux. Technical Report TR-2001-4, Carnegie Mellon University, Feb. 2002.
11. How to Use TCPCP2. http://prdownloads.sourceforge.net/tcpcp2/how_to_use.pdf?download
12. Open Source Development Labs, Carrier Grade Linux Requirements Definition Documents V3.2. Feb. 2006.
13. W. Almesberger. TCP Connection Passing. In Proceedings of Ottawa Linux Symposium 2004, vol. 1, pp. 9–21, July 2004.
14. F. Leite. Load-Balancing HA clusters with No Single Point of Failure. In Proceedings of the 9th International Linux System Technology Conference (Linux-Congress 2002), pp. 122–131, Sep. 2002. http://www.linux-kongress.org/2002/papers/lk2002-leite.html
15. T. Ikebe, Y. Kawarakaki, and J. Yamanaka. Practical TCP Session Take-over Method for High-availability Network Service. In Proceedings of 6th Asia-Pacific Symposium on

Information and Telecommunication Technologies, 2005 (APSITT 2005), pp. 1–6, Nov. 2005.
16. J. Mahdavi and S. Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Technical note sent to the end2end-interest mailing list, January 1997.