客戶網路管理雛型系統 CORBA 介面設計

徐祖詒　邱萬德　曾宇仲　王承蔭
中華電信研究所　交換技術研究室
wander@ms.chttl.com.tw

摘　要

本文發表一客戶網路管理(Customer Network Management, CNM)系統雛型，經由開放一個窗口，讓客戶能直接上網查詢公眾網路即時狀況。客戶網路管理可提供組態管理(Configuration Management、效能管理(Performance Management)、障礙管理(Fault Management)、安全管理(Security Management)以及帳務管理(Accounting Management)等功能。本雛型以 ATM 網管系統為例，並採用 CORBA(Common Object Request Broker Architecture)技術開發，伺服端(server)使用 C++ 語言撰寫，客戶端(client)則採用 Java 語言開發。本文詳述了如何設計伺服端與客戶端之間的溝通介面—CNM IDL (Interface Definition Language)以及伺服端與客戶端實作 IDL 時應注意之事項。另外，在 CORB 開放架構下，本文亦加入了使用者帳號登入，以確保用戶與電信業者之資訊安全與保密性。本系統架構，除了提供遠端網路管理的功能之外，未來，可直接擴充以提供電子商務方面之服務。

關鍵詞：客戶網路管理、Customer Network Management、CORBA、電子商務、Java

1 客戶網路管理(CNM)

由於 Internet 的普及，帶來了無限的方便與商機。愈來愈多的企業，計畫建置一個屬於企業本身的企業網路，以提升企業的競爭力與服務品質。若是一個跨國企業欲建置一個全球性的企業網路，企業本身鋪設光纖或是數據專線，成本勢必太高。為了節省成本，最直接的方法就是，向現有電信業者承租固定頻寬之數據專線；但往往數據專線斷線，企業並不能即時地全盤掌握整個承租專線的網路狀況。

擁有公眾網路的電信業者，必須提供一個窗口，讓承租的客戶能夠即時查詢所承租公眾網路的情形，如果可以和客戶自己的網管系統連線，成效必然更好。一旦網路發生障礙，可以協助客戶找出障礙所在，釐清責任分界點。電信業者，亦可藉由 CNM 服務，讓客戶監督網路品質，進而提升網路營運的穩定性及客戶滿意度。

商用客戶網路管理架構 [1,2]

商用客戶網路管理系統，如圖 1；在公眾網路網管中心方面，必須和商務管理系統(Business Management System)、服務管理系統(Service Management System)與網路管理系統(Network Management System)相結合，向客戶提供一個完整服務的窗口—CNM Agent。客戶可利用 CNM Agent 得到之完整資訊，自行開發應用程式(Application)，整合至客戶本身之網管系統(NM)或商務管理系統(BM)。

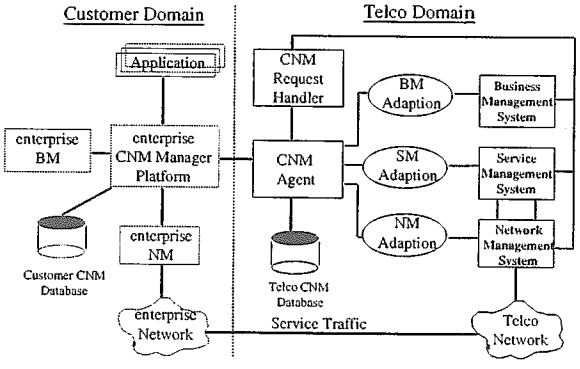

圖 1　商用客戶網路管理架構

商用客戶網路管理系統，主要提供以下五大管理功能：

(1) 組態管理(Configuration Management)：

線上查詢即時網路狀況、線上承租新服務及重新調整所承租之網路或服務組態。

(2) 障礙管理(Fault Management)：

報告、追蹤障礙及縮小障礙範圍，並提供排除障礙處理流程以及確認障礙發生所在。

(3) 效能管理(Performance Management)：

監控網路 QoS，如 throughput、delay 及 availability 等，產生訊務報表以確認是否符合合約內容。

(4) 帳務管理(Accounting Management)：

提供客戶其所承租網路之使用量統計報表，以提供客戶提報下一年度網路預算之參考。

(5) 安全管理(Security Management)：

使用者管制與認證，並區隔用戶與用戶之間以及用戶與電信業者之間的資訊，以確保用戶與電信業者資訊安全與保密性。

2 CORBA 簡介

OMG (Object Management Group)是一個由會員贊助而成立的非營利組織，其目的在於推廣物件導向(Object-Oriented)的觀念及使用，並致力於加強軟體的可攜性(portability)、再利用性(reusabilit)以及互通性(interoperability)。CORBA (Common Object Request Broker Architecture)為OMG在一九九一年十二月提出之物件導向分散式工作環境規格，目前最新版本為一九九八年二月之 CORBA/IIOP 2．版，互通性及C＋＋與Java對應(mapping)之標準，均於新版之標準中，更明確地制訂。

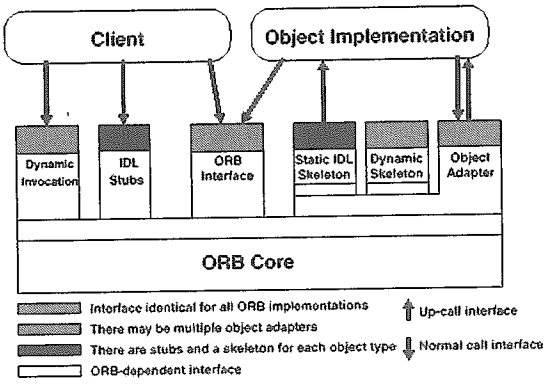

圖 2　CORB　介面結構

傳統分散式架構，如 DCE(Distributed Computing Environment)，並沒有類別或繼承之觀念及功能，CORBA規格最大的特色即為物件導向觀念的導入。為了建構一個分散式應用程式環境之標準，OMG 提出了OMA(Object Management Architecture)架構[4,5,6]，其核心部分稱之為 ORB(Object Request Broker)，專司物件所在位置透明化之找尋、物件的啟動及物件間之溝通[7]，如圖 2。當客戶端呼叫一個運算，ORB 負責搜尋物件實作 (Object Implementation)，必要時，啟動物件實作，傳送請求給物件，並且傳回結果給呼叫者。

3 CNM CORBA 介面系統架構

本系統採用 Three-Tier 之系統架構，前端為瀏覽器，後端則是網管系統，如圖 3。連結瀏覽器與網管系統則是一個 CNM 服務的仲介者(CNM Servant)，同時，亦具有網頁伺服器(Web Server)之功能。此仲介者，處理來自於瀏覽器的使用者要求；經過處理後，再傳回瀏覽器。必要時，仲介者會向後端網管中心取得即時的網路狀況資訊。

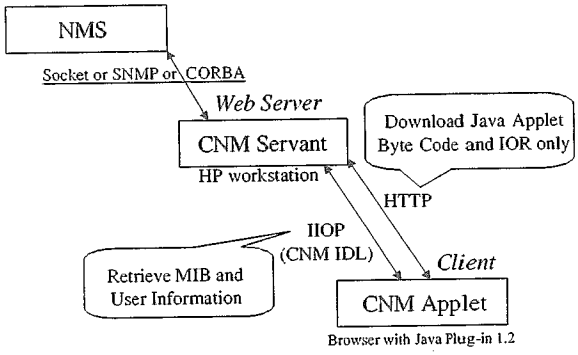

圖 3　CNM CORB　介面系統架構

前端的瀏覽器，可以是常見的 Netscape 通訊家或是 IE 探險員，另外，仍須安裝 Java Run Time Environment 1.2。後端則是任何一種網管系統，必須要 有 提 API 或是一擷取資訊之窗口，以供外界擷取即時網管資訊。舉例來說，Cisco 交換機之網管系統 StrataView Plus，有提供一個窗口 SNMP Proxy Agent，透過 SNMP v1 協定，可擷取交換機即時網路資訊。中間網頁伺服器，則採用 Apache Web Server。至於，CNM 服務仲介者及使用者操作介面(HMI)設計，分別詳述於第五節及第六節。

一開始，使用者輸入 CNM 服務網址(URL)之後，瀏覽器即向 Web Server 讀取 CNM 服務首頁。瀏覽器解讀此首頁，發現存在一 Applet Tag，隨即下載經壓縮過之 Java

Applet Bytecode—CNMApplet.jar，並啟動 Java Plug-in 1.2(JRE)。接下來，JRE 解壓縮 CNMApplet.jar 檔案，並執行此 Applet。此時，即出現一登入畫面，要求使用者輸入帳號及密碼。使用者輸入其帳號與密碼後，即可登入系統。

當使用者登入系統時，Java Applet 會先向 Web Serve 取得登入程序之 IOR。ORB 啟動後，根據此登入程序 IOR，向 CNM 服務仲介者(CNM Servant)進行使用者認證。若認證成功，即真正進入客戶網路管理系統。取得 IOR 之後，Applet 便使用 IIOP 通訊協定和仲介者(CNM Servant)溝通傳遞訊息，並要求仲介者提供服務；下載網頁及 Java Applet Bytecod 與下載 IOR，則是運用 HTT 通訊協定。

4 CNM IID 設計原則

IDL (Interface Definition Language) 類似 C＋＋語法，又與 C＋＋略有不同。IDL 僅定義伺服端(server)與客戶端(client 之間的溝通介面—資料結構與物件類別(class)以及其可被啟動(invoke)之方法(method)或運算(operation)集合。對於如何實作這些物件類別及其方法，完全不予規範。程式發展者可自由選擇適當的程式語言來實現。

```
module cnm {
  /*enum AtmOperStatusTyp 、AtmAdminStatusTyp 、
  RowStatus、AtmTrafficDescrType 與 struct vplStruct、
  trafficStruct、xconnectStruct 以及 Interface cm、fm、pm、
  sm 、am 、atmTrafficDescrParamEntry、atmVplEntry、
  atmVpCrossConnectEntry 定義於此*/
  interface cnmService {
     struct userStruct {
        string val_coName;
        unsigned short val_noPVC;
        string val_info;
     };
     userStruct userInfo();
     cm cmObj();
     fm fmObj();
     pm pmObj();
     sm smObj();
     am amObj();
  };
  interface loginService {
     cnmService login(in string username, in string
                                          password);
  };
};
```

表 1 CNM IDL 精簡列表

在 CNM IDL 只有一個 cnm 模組(module)，包含了兩個最重要的介面(interface)：loginService 以及 cnmService。介面 loginService，主要是作安全管理以及預留將來收費的空間，如圖 4。介面 loginServic ，僅包含一個方法 login。客戶端啟動方法 login 時，必須輸入使用者帳號名稱以及密碼。若認證失敗，login 傳回 NUL 值：若認證成功，則傳回類別為 cnmService 之實例參考(Instance Reference)。值得注意的是，方法 login 傳回每一位使用者專屬之 cnmService 實例參考，不同的使用者，有著不同的實例參考。這些實例參考，為同一種類別 cnmService，但是，是不同的實例(Object Instance)。

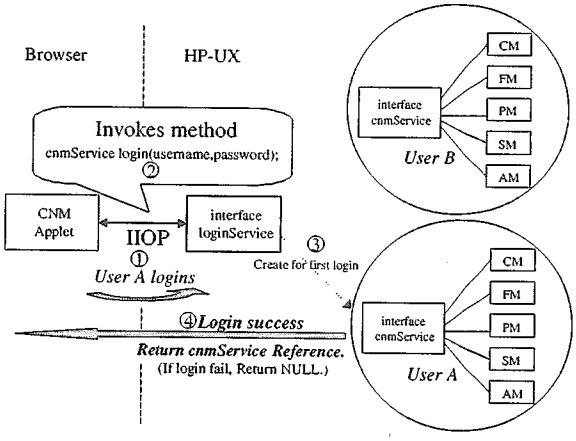

圖 4 客戶網路管理系統安全機制

介面 login，除了作使用者安全認證外，仍必須負責動態配置此使用者專屬之 cnmService，並分別起始 cm、fm、pm、sm 及 am 等 interface 之實例(Object Instance)。至於 cm、fm、pm、sm 及 am interface，吾將稍後陸續介紹，並且詳列於附錄之中。未來，可將計費機制置於介面 login，採計次計費。

介面 cnmService 包含所有相關於此使用者需要之網路狀態資訊，如組態管理(Configuration Man agement)、障礙管理 (Fault Management) 、效能管理 (Performance

Management)。除了必要的網路管理功能外，亦增加了安全管理(Security Management)、帳務管理(Accounting Management)等功能以及系統改版相關的最新資訊。

網路管理方面：

有關介面 cm，吾人直接從 RFC 1695 擷取 PVC (Permanent Virtual Connection)部分的 MIB sub-tree，重要的表格包括 ifTable、atmVpCrossConnectTable、atmVplTable、atmTrafficDescrParamTable 等等[3]。介面 pm 包含了一些交換機性能相關的參數。介面 fm 則包含了障礙申告與查詢，以查詢障礙處理情形。

(1) 組態管理(Configuration Management)

介面 cm 的所有方法，可分成兩類：

一類方法為所有元素以物件型態一次傳回：

xconnectEntrys xconnectObjs()、vplEntrys vplObjs()、trafficEntrys trafficObjs()

另一類方法為所有元素以結構型態一次傳回（以斜體表示）：

xconnectStructs allXconnect()、vplStructs allVpl()、trafficStructs allTraffic()

```
interface cm {
  typedef          sequence<atmTrafficDescrParamEntry>
                                              trafficEntrys;
  typedef sequence<trafficStruct> trafficStructs;
  typedef sequence<atmVplEntry> vplEntrys;
  typedef sequence<vplStruct> vplStructs;
  typedef          sequence<atmVpCrossConnectEntry>
                                              xconnectEntrys;
  typedef sequence<xconnectStruct> xconnectStructs;
  //Get All Objet Instances
  trafficEntrys trafficObjs();
  vplEntrys vplObjs();
  xconnectEntrys xconnectObjs();
  //Get All Structures: xconnectStructs, vplStructs and
  trafficStructs
  trafficStructs allTraffic();
  vplStructs allVpl();
  xconnectStructs allXconnect();
};
```

為何要提供兩組類似的方法，供客戶端啟動呢？ 提供物件型態，可應用於 MIB Browser，允許客戶端一次只讀取一個欄位值，類似原本在 SNMP 當中，給定特定的 MIB Object ID，向 Agent 讀取欄位值；提供結構型態，允許客戶端可一次讀取整個表格。

結構 xconnectStruct 以及介面 atmVpCrossConnectEntry 皆是根據 RFC 1695 中定義的 AtmVpCrossConnectEntry，直接轉譯而成[3]。是故，兩者皆為 atmVpCrossConnectTabl 中的某一列，所不同的是一個是結構形式，另一個為物件形式。

資料型態 xconnectEntrys、xconnectStructs 皆被定義成一個未定長度的陣列。xconnectEntrys 陣列中的每一個元素都是 atmVpCrossConnectEntry 物件；xconnectStructs 陣列中的每一個元素都是 xconnectStruct 結構。

介面 atmVpCrossConnectEntr，僅提供一個方法 getXconnect，將 xconnectStruct 結構一次取回。吾人將 atmVpCrossConnectTable 表格中的每一列變成一個、一個類別為 atmVpCrossConnectEntr 的物件實例(Object Instance)。每一個 atmVpCrossConnectEntry Instance，代表 atmVpCrossConnectTabl 表格中的某一個列。換句話說，方法 getXconnect 僅能夠一次讀取 atmVpCrossConnectTable 表格中的其中某一列。

介面 atmVpCrossConnectEntry 中，關鍵字 attribute，IDL 編譯器將會分別編譯成兩個方法，一個是 set 方法，另一個是 get 方法，如 attribute unsigned long atmVpCrossConnectIndex：

set 方法：

```
virtual void atmVpCrossConnectIndex(CORBA::Ulong
                                              value);
```

get 方法：

```
virtual CORBA::ULong atmVpCrossConnectLowIfIndex();
```

關鍵字 readonly attribute，IDL 編譯器僅編譯成一個 get 方法，如 readonly attribute string atmVpCrossConnectName：

get 方法：virtual char* atmVpCrossConnectName();

是故，客戶端模組可一次讀取一個欄位或是一次設定一個欄位，以應用於 MIB Browser。

```
struct xconnectStruct {
  string val_atmVpCrossConnectName;
  string val_atmVpCrossConnectLowSwitchName;
  string val_atmVpCrossConnectHighSwitchName;
  unsigned long val_atmVpCrossConnectIndex;
  unsigned long val_atmVpCrossConnectLowIfIndex;
  unsigned short val_atmVpCrossConnectLowVpi;
  unsigned long val_atmVpCrossConnectHighIfIndex;
  unsigned short val_atmVpCrossConnectHighVpi;
  AtmAdminStatusType
                val_atmVpCrossConnectAdminStatus;
  AtmOperStatusType
                val_atmVpCrossConnectL2HOperStatus;
  AtmOperStatusType
                val_atmVpCrossConnectH2LOperStatus;
  unsigned long val_atmVpCrossConnectL2HLastChange;
  unsigned long val_atmVpCrossConnectH2LLastChange;
  RowStatus val_atmVpCrossConnectRowStatus;
};

typedef          sequence<atmVpCrossConnectEntry>
                                    xconnectEntrys;
typedef sequence<xconnectStruct> xconnectStructs;

interface atmVpCrossConnectEntry {
  readonly attribute string atmVpCrossConnectName;
  attribute unsigned long atmVpCrossConnectIndex;
  attribute unsigned long atmVpCrossConnectLowIfIndex;
  attribute unsigned short atmVpCrossConnectLowVpi;
  attribute unsigned long atmVpCrossConnectHighIfIndex;
  attribute unsigned short atmVpCrossConnectHighVpi;
  attribute                    AtmAdminStatusTyp
                    atmVpCrossConnectAdminStatus;
  readonly attribute AtmOperStatusType
                    atmVpCrossConnectL2HOperStatus;
  readonly attribute AtmOperStatusType
                    atmVpCrossConnectH2LOperStatus;
  readonly attribute unsigned long
                    atmVpCrossConnectL2HLastChange;
  readonly attribute unsigned long
                    atmVpCro sConnectH2LLastChange;
  attribute RowStatus atmVpCrossConnectRowStatus;
  //get Virtual Path Cross Connect table entry once
  xconnectStruct getXconnect();
};
```

(2) 障礙管理(Fault Management)

介面 fm，定義了 troubleStruct 結構，亦定義了兩個方法 petition、getTroubles。方法 petition，可讓使用者直接報告障礙，讓公眾網路網管中心知曉。方法 getTroubles，則提供客戶端直接查詢過去曾發生之障礙及其處理情形。

```
interface fm {
  struct troubleStruct {
     string<60> request;
     string<60> status;
  };
  typedef sequence<troubleStruct> troubles;
  boolean petition(in string<60> trouble);
  troubles getTroubles();
};
```

(3) 效能管理(Performance Management)

介面 pm，定義了一個結構 vplPmStruct，包含了 VPL 的名稱，以及 VPL 自上次重新啟動所丟棄的細胞總數與所通過的細胞總數。介面 pm，提供了一個方法 getVplPms，可將所有 VPL 之性能參數，從伺服端一次傳回至客戶端。所有交換機性能相關參數，皆可定義於此，如 PVC 之訊務參數。

```
interface pm {
  struct vplPmStruct {
      string          atmVplName;
      unsigned long   cellsDiscarded;
      unsigned long   cellsPassed;   /*discarded by UPC*/
  };
  typedef sequence<vplPmStruct> vplPmStructs;
  vplPmStructs getVplPms();
};
```

安全帳務管理方面：

安全管理，目前僅提供密碼變更服務；帳務管理，則提供帳單服務，如查詢每月通信費支出等。

(1) 安全管理(Security Management)

介面 sm，僅提供一個方法 change，可供使用者更改密碼。若更改成功，傳回 true；若更改失敗，則傳回 false。

```
interface sm {
  boolean change(in string username, in string oldPass, in
                                          string newPass);
};
```

(2) 帳務管理(Accounting Management)

介面 am，僅提供一個方法 getAccount，供客戶端查詢用戶帳單資訊。所有用戶帳務資訊皆可定義於 struct record 當中，讓客戶端可一次讀取。

```
interface am {
  struct record{
      string startDate;
      string rate;
  };
  record getAccount();
};
```

在本文中，伺服端採用 C++語言實作，客戶端則運用 JAVA 語言實作，接下來的兩節，將分別詳述伺服端及客戶端實作時，應注意之細節。

5 CN 服務仲介者(CNM Servant)模組介紹

CNM 服務仲介者，採用 MICO CORB ORB[9]，並將 CNM IDL 編譯成 C++語言，以 C＋＋語言實作所有 IDL 所定義之 interface。

主程式說明如下：

```
int main( int argc, char *argv[] ) {
  //起始ORB
  CORBA::ORB_var orb =CORBA::ORB_init( argc, argv,
  "mico-local-orb" );
  CORBA::BOA_var boa = orb->BOA_init(argc, argv,
  "mico-local-boa");
  //起始loginService_impl，並將其IOR寫入檔案中
  //Client端利用Web Server，將IOR經HTTP傳回Client
  端
  loginService_impl* obj = new loginService_impl();
  CORBA::String_var ref = orb->object_to_string( obj );
  ofstream out1 ("../../WWW/prototype/cnmid");
  out1 << ref << endl;
  out1.close();
  boa->impl_is_read
  (CORBA::ImplementationDef::_nil());
  orb->run();
  //此時，可服務客戶端之要求
  CORBA::release(obj);
  return 0;
}
```

表 2 CNM Servant 主程式列表

6 客戶端(Client)模組介紹

使用者操作介面，開發於 Java II 平台，也就是 Java Development Kit 1.2[11]。Java II 正式發表於一九九八年十二月初。相較於 JDK 1.1.x，主要美化了使用者介面元件 Swing，以及簡化了事件(Event)處理之派工法則(dispatch)。至於客戶端之 ORB，則先後測試過 Visibroker ORB 以及 SunSoft 所開發之 ORB(內含於 Java II)。

客戶端模組主要類別及其功能，詳列如下：

(1) CNMApplet：使用者操作介面、定義所有使用者介面

元件之事件處理方式。

(2) CNMMenuBar：客戶網路管理系統功能，包含登出、切換視窗顯示方式、啟閉浮貼說明標籤以及系統操作說明。

(3) CNMORB：負責Applet向仲介者或伺服端擷取網管訊息以及使用者資訊。

(4) Retriever：透過CNMORB，向仲介者或伺服端擷取網管資訊，並加以分析，呈現於Applet之上。

(5) CM、FM、PM、AM、SM Panel：所有管理功能，皆各自獨立成一個Panel，集中放置於Tabbed Pane之中。

(6) CrossConnectFrameClass、TrafficFrameClass與VPLFrameClass：負責顯示CrossConnect、Traffic、VPL表格。

(7) MIB瀏覽器(Browser)：類似SNMP MIB Browser之功能。

CNMORB為整個客戶端模組中，唯一向CNM服務仲介者(CNM Servant)溝通的類別，並撰寫成Java Abstract Class。因此客戶端模組之其他的類別如MIB Browser、CrossConnectFrameClass、VPLFrameClass、Retriever、TrafficFrameClas 等，皆是透過CNMORB，擷取CM、FM、PM、AM資訊以及線上即時更改密碼。

CNMORB主要作法說明如下：

```
try {
 //起始ORB
 orb=org.omg.CORBA.ORB.init
                    (CNMConstant.thisJApplet,null);
 //讀取loginService IOR - CNM Servant Reference
 String
 URL cnmid=new
   URL("http://xxx.xxx.xxx.xxx/~user/prototype/cnmid");
 BufferedReader in = new BufferedReader(new
            InputStreamReader(cnmid.openStream()));
 String loginServiceObjRefStr=in.readLine();
 //將IOR轉換成loginService物件
 org.omg.CORBA.Object loginServiceTemplate =
         orb.string_to_object(loginServiceObjRefStr);
 loginServiceObj=cnm.loginServiceHelper.narrow
                                    (loginServiceTemplate);
 //此時，即可直接呼叫方法 login
} catch (Exception e) {
 System.out.println("ERROR(init ORB): " + e) ;
 e.printStackTrace(System.out);
}
```

表 3　CNMORB 精簡列表

客戶管理系統登入畫面如圖 5：

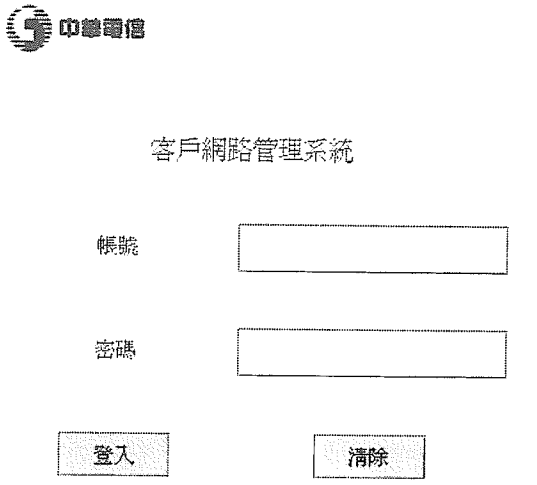

圖 5　客戶網路管理系統登入畫面

系統操作畫面如圖 6：

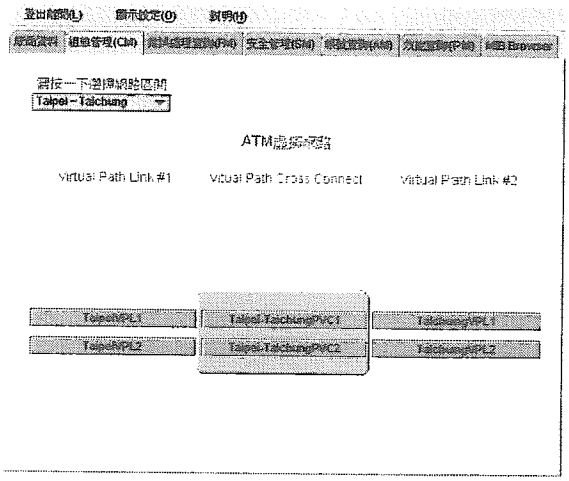

圖 6　客戶管理系統操作畫面

7 未來發展方向

本文僅提供遠端網路管理功能，將來可以既有的系統架構，額外提供電子商務服務，包括線上訂購服務、信用卡轉帳服務、網路使用量統計分析等等。同時，在本文中，CNM 服務仲介者(CNM Servant)與 CNM Java Applet

皆透過 IIOP 彼此溝通，並以明碼傳遞。未來，可考慮將類似 SS 安全機制加入，以增加電子商務之安全性。

8 結論

本文實際應用 CORB 技術於客戶網路管理系統，證明 CORB 可應用於客戶服務系統，充分發揮了分散式運算環境的優點。本文詳細探討伺服端與客戶端之間的介面與實作的細節，同時，在 CORB 開放架構下，加入了使用者帳號登入，以確保用戶及電信業者之資訊安全與保密性。

9 誌謝

對於參與 CNM 研發工作之所有同仁的同心協力，俾使本系統能順利完成，在此一併致謝。

參考文獻

[1] af-nm-0019.000, Revision 1.04, "Customer Network Management (CNM) for ATM Public Network Service (M3 Specification)," The ATM Forum Technical Committee, October, 1994

[2] G. Holliman, M. Hinchliffe, Nigel Cook and Peter Barnes, "Customer Network Management," http://www.citr.com.au/02.TechnicalJournal/01.Papers/P4_CNM.html

[3] Internet Engineering Task Force RFC 1695, M. Ahmed and K. Tesink, "Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2," Bell Communications Research, August 1994.

[4] Object Management Group, Object Management Architecture Guide, Revised 1995.

[5] Object Management Group, CORBA: Common Object Request Broker Architecture and Specification Revision 2.0, July 1995.

[6] Object Management Group, CORBA Services: Common Object Services Specification, March 1995.

[7] Deron Liang, Winston Lo, and Shyan -Ming Yuan, "Introduction to CORBA 2.0 Specification and its Future Trends," Proceeding of 1996 Workshop on Distributed System Technology and Applications, pp. 3-10, May 1996.

[8] Andreas Vogel and Keith Duddy, Java Programming with CORBA, John Wiley & Sons, Inc, 1997.

[9] Kay Romer and Arno Puder, "MICO is CORBA," http://www.vsb.cs.uni-frankfurt.de/~mico/doc/doc.html

[10] Java Plug-in 1.2, http://java.sun.com/products/plugin/index.html

[11] Java development kit Version 1.2, http://java.sun.com/products/jdk/1.2/

[12] Patrick Chan, The Java Developers Almanac 199 9 (Java Series), January 1999.