第二章　文獻探討

2.1 Web Service 技術

Curbera[6]等人之研究中提到 Web Service 主要包含 XML、SOAP、WSDL、UDDI 等四種標準。這些都是開放式的標準，因此網際網路中不同的平台之間可以透過 Web Service 做溝通，改變傳統上人與應用程式之間溝通的模式，變為應用程式與應用程式之間或者電腦與電腦之間的溝通模式。舉例來說：傳統的人與應用程式間溝通就像人開瀏覽器，輸入需要的網址跟主機要求網頁資料；Web Service 溝通的模式變為應用程式間互相傳遞訊息，就像客戶端程式跟伺服端程式預訂一張機票，伺服端程式回傳預訂的結果。根據 W3C 對 Service 的定義[7]，一個 Service 會使用 WSDL 定義，而 Web Service 在 W3C Working Group 的定義如下：

A Web service is a software system identified by a URI [8], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

由此定義可見 Web Service 基於 XML-based 的訊息傳遞方式，建立在網路的既有標準上，但不要求依附在特定的通訊協定或特定的資訊平台上。目前實作上

以跑在 HTTP 通訊協定上居多，為 SOA(Service Oriented Architecture)模式之應用。

2.1.1 XML(Extensible Markup Language)

Weitzel 等人表示已有越來越多廠商支援 XML 格式，應用範圍包含 Application Server、Client 與 Storage[9]，XML 為開放的格式，可以在不同平台上使用，互相溝通。

根據 W3C 對於 XML 之定義[10]，XML 是個簡單可格式化的標記語言，可以自己定義，發展出基於 XML 的標準。XML 因此被視為網際網路與商業應用的關鍵技術，目前有很多基於 XML 的技術目前正持續發展中。

Barillot 與 Achard 認為使用 XML 來做資料交換，具有以下四個優點[11]：

1. XML 為公開的標準，具有開放的特性。
2. XML 是標籤語言，可以自行定義所需要的格式，可被符合定義的應用程式在不同平台中解釋，不會有平台的限制。
3. XML 能使用在不同的應用程式，可使用於不同領域的地方。
4. 有定義 DTD(Document Type Definition)的功能，可用來定義規劃 XML 檔案中包含元素的屬性，達成資料結構的統一。

XML 為純文字的檔案，不會像 Word 的檔案在不同的 Word 版本中，可能會有相容性的問題，因此可以長時間儲存資料，並可依照需要作不同的呈現。目前根據 XML 延伸出來的標準非常的多，SVG、MathML、ebXML…等都是。

一個簡單的 XML 檔案可以如下【表二】所示，personData 是 root，檔案中包含兩筆資料，第一筆資料的姓名是 Alice、電話 0911123456、生日 08/16/1981；第二筆資料的姓名是 Bob、電話 0944000000、生日 07/08/1982。

【表二】XML 範例

```
<?xml version="1.0" encoding="utf-8"?>
<personData>
    <name>Alice</name>
    <phone>0911123456</phone>
    <birth>08/16/1981</birth>
    <name>Bob</name>
    <phone>0944000000</phone>
    <birth>07/08/1982</birth>
</personData>
```

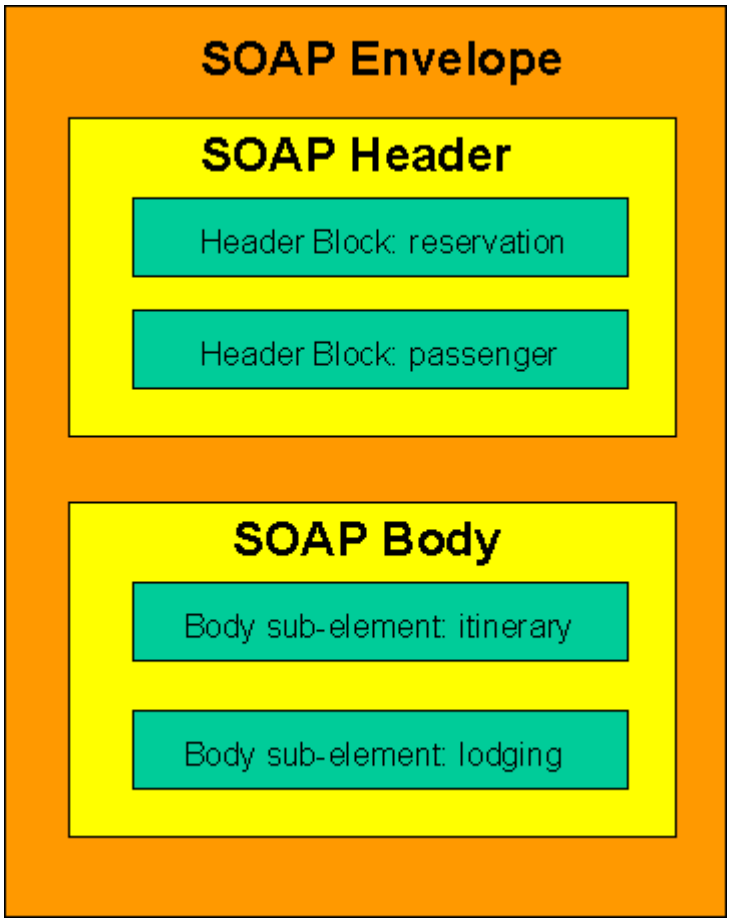
2.1.2 SOAP(Simple Object Access Protocol)

SOAP(Simple Object Access Protocol)，且其底層是以 XML 檔案為基礎。因為 SOAP 本身沒有規定其通訊協定，所以可利用此特性來整合不同平台上的“遠端呼叫”。SOAP 的組成有 3 大部分[12]如下【圖五】，主要組成有三部份，三個部份分述如下：

1. Envelope：SOAP Envelope 是一份 XML 文件，它有兩個子元素、分別是 Header 及 Body。

2. Header：用來寫入與應用程式無關的資訊。

3. Body：包含應用程式專屬的資訊、如輸出及輸入等資訊。



【圖五】 SOAP 組成三部分
〔資料來源：http://www.w3c.org〕

簡單的 SOAP 範例如下【表三】所示：

【表三】SOAP 範例

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                 env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                 env:mustUnderstand="true">
   <n:name>My Name</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
```

```
      <p:departureDate>2001-12-14</p:departureDate>
      <p:departureTime>late afternoon</p:departureTime>
      <p:seatPreference>aisle</p:seatPreference>
    </p:departure>
    <p:return>
      <p:departing>Los Angeles</p:departing>
      <p:arriving>New York</p:arriving>
      <p:departureDate>2001-12-20</p:departureDate>
      <p:departureTime>mid-morning</p:departureTime>
      <p:seatPreference/>
    </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

2.1.3 WSDL(Web Service Description Language)

根據 W3C 的規範說明[13]，WSDL 為 XML 格式用來描述服務與服務傳遞的訊息資訊，一個簡單的 WSDL 檔案如下【表四】所示：

【表四】WSDL 範例

```
<?xml version="1.0"?>
<definitions xmlns:tns="http://example.com/stockquote.wsdl" xmlns:xsd1="http://example.com/stockquote.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://example.com/stockquote.wsdl"
name="StockQuote">
    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="TradePriceRequest">
                <complexType>
                    <all>
                        <element name="tickerSymbol" type="string"/>
                    </all>
                </complexType>
            </element>
            <element name="TradePrice">
                <complexType>
                    <all>
                        <element name="price" type="float"/>
                    </all>
                </complexType>
            </element>
        </schema>
    </types>
    <message name="GetLastTradePriceInput">
```

```
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>
    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
            <input message="tns:GetLastTradePriceInput"/>
            <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
            <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
            <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>
</definitions>
```

1. types：定義使用的資料型態

2. Message：為抽象型別，定義通訊的資料訊息名稱與資料型態(types)

3. Operation：為抽象的描述，敘述此服務提供的功能名稱

4. Port Type：為 Operation 的集合，定義此服務提供哪些 Operation

5. Binding：定義將 Port Type 跑在何種通訊協定之上

6. Port：指定 Binding 需要的服務通訊之端點位址

7. Service：為多個 Port 的集合，可指定如何存取服務之資訊

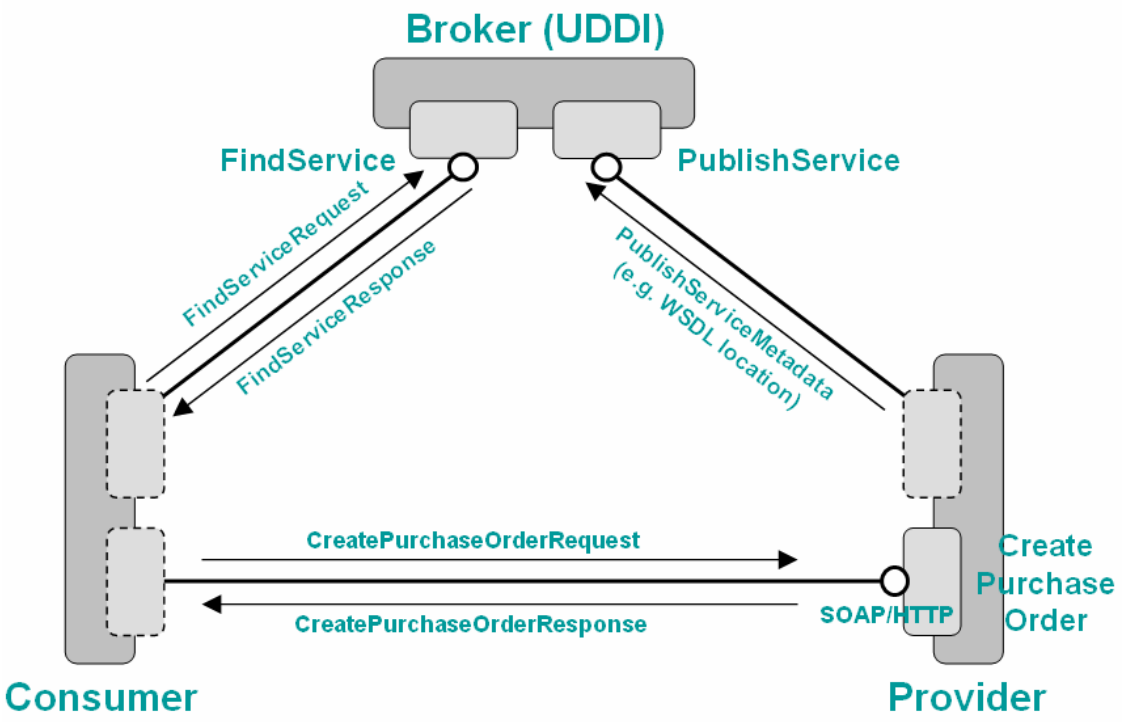
目前主要實作 Web Service 的平台，大都有 WSDL 自動產生的功能，主要描

述 Web Service 的存取介面，其功能是為了讓需要的應用程式可以做連結。

2.1.4 UDDI(Universal Description Discovery and Integration)

目前 UDDI 由 OASIS TC(http://www.oasis-open.org)主導，主要是讓 Web Service 提供者登錄提供的服務，讓需要的程式可以查詢到。Microsoft、IBM、SAP 三家公司則依此標準共同維護及管理全球商業 UDDI Registry(Universal Bussiness Registry，簡稱 UBR)，提供 UDDI 在商業上的應用。

透過 UDDI 的機制，可提供使用者在線上以電子式的方式直接找到需要的服務在哪邊可以取得，有點類似我們可以透過電話簿找到公司的電話，其在 Web Service 扮演的角色如下【圖六】所示：



【圖六】 UDDI 在 Web Service 扮演之角色
（資料來源：http://www.w3.org/2004/10/presentations/claus.ppt）

服務提供者平時將提供的服務登錄於 UDDI Server 上，當服務使用者要使用時，服務使用者一開始不知道服務提供者在哪裡，因此向 UDDI Server 要求需要的服務，UDDI Server 回覆服務提供者的 URI，接著才向服務提供者要求服務。UDDI 扮演路標的角色，讓需要服務的使用者可以找到提供服務的地方。

2.2 IPv6 技術

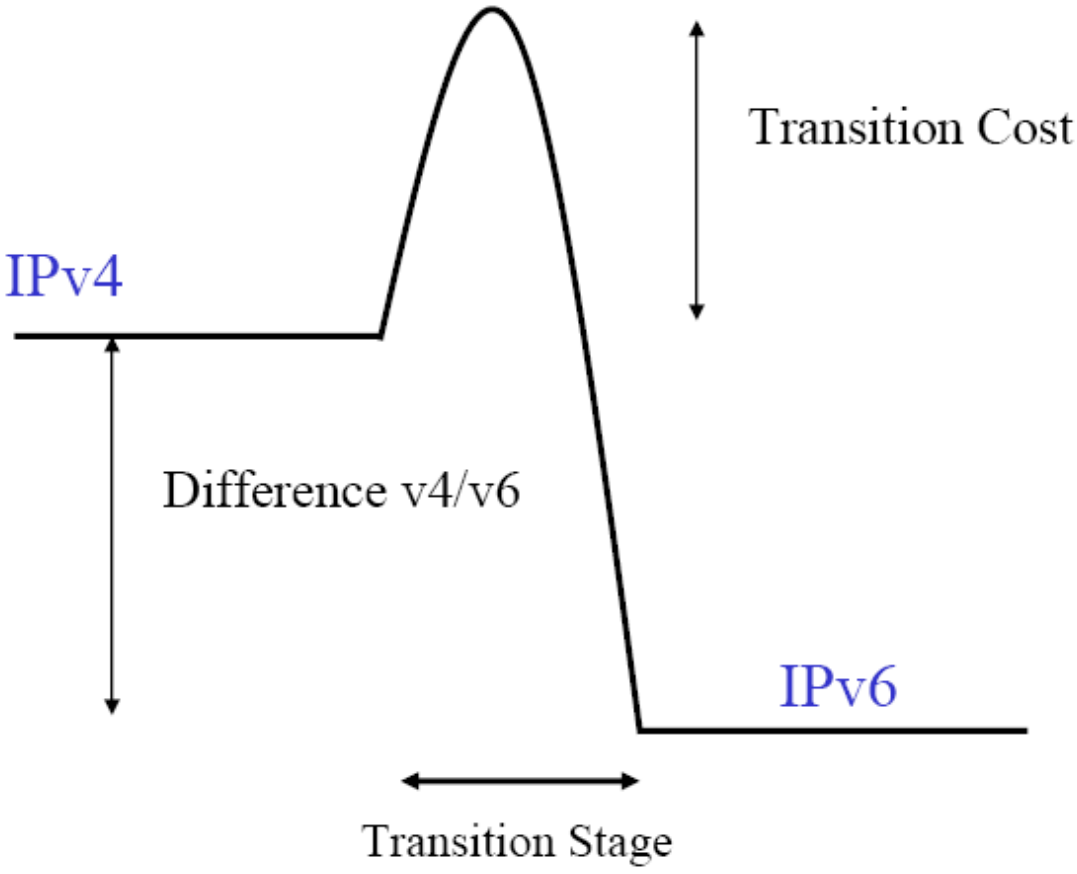
隨著電腦網際網路使用日益頻繁，傳統的 IPv4 的 IP 位址(例如：140.122.76.111)與功能已經漸漸不敷使用，目前很多國家都積極成立 IPv6 的計畫，要慢慢先改成使用 IPv4 與 IPv6 共存的網路環境，接著再慢慢的改成 IPv6 的網路環境。IPv4 位址長度為 32 bit，IPv6[14]位址長度則增加為 128bit，並增加了很多新的功能，新增加的功能以 IP 層內附的安全性與移動性最為重要，並有支援 IPv6 位址自動取得的功能[15]，達到網路隨插即用，不需要其他複雜的設定即可連上網路。一般狀況而言 IPv6 可解決的問題包括：

1. 能夠迅速繼承 IPv4 之特性與多元應用服務
2. 可避免使用 NAT 技術造成網路效能下降
3. IP 位址空間充足(128 位元)
4. 降低網路運作之複雜度
5. 可解決利用創新通訊技術所帶來的限制
6. 可提供 IP 層內建的網路安全支援(IPv6 Security)
7. 網路品質服務(QoS)支援度較高

8. 內建移動性功能(Mobile IPv6)

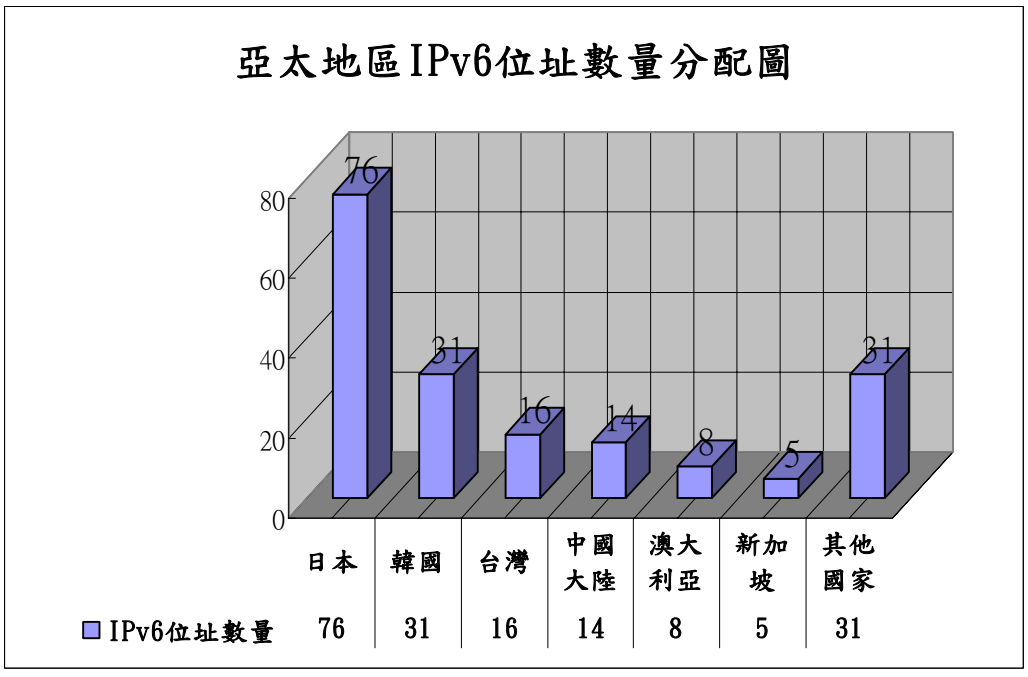
在 IPv6 安全性的部份，在 IP 層內建網路安全性的功能，支援 AH[16]與 ESP[17]等安全協定，兩者一起配合，確保網路封包可以安全的傳送，不會被竊聽與遭到修改。在移動性方面，在 IP 內建之移動性支援[18]，配合 IPv6 鄰居主機查找[19]與 IPv6 位址自動設定[20]技術，實現 IPv6 在移動性網路中的使用。

由於現階段大眾使用 IPv4 協定，若全面性將 IPv4 轉換為 IPv6 在初期需要耗費較高的成本，然而預期 IPv6 的優越特性將帶來較高之成本節約效應，如下【圖七】表示從 IPv4 轉換至 IPv6 之系統運作成本趨勢，轉換成 IPv6 之後，Cost 可以下降很多，因此使用 IPv6 技術值得投資與研究。

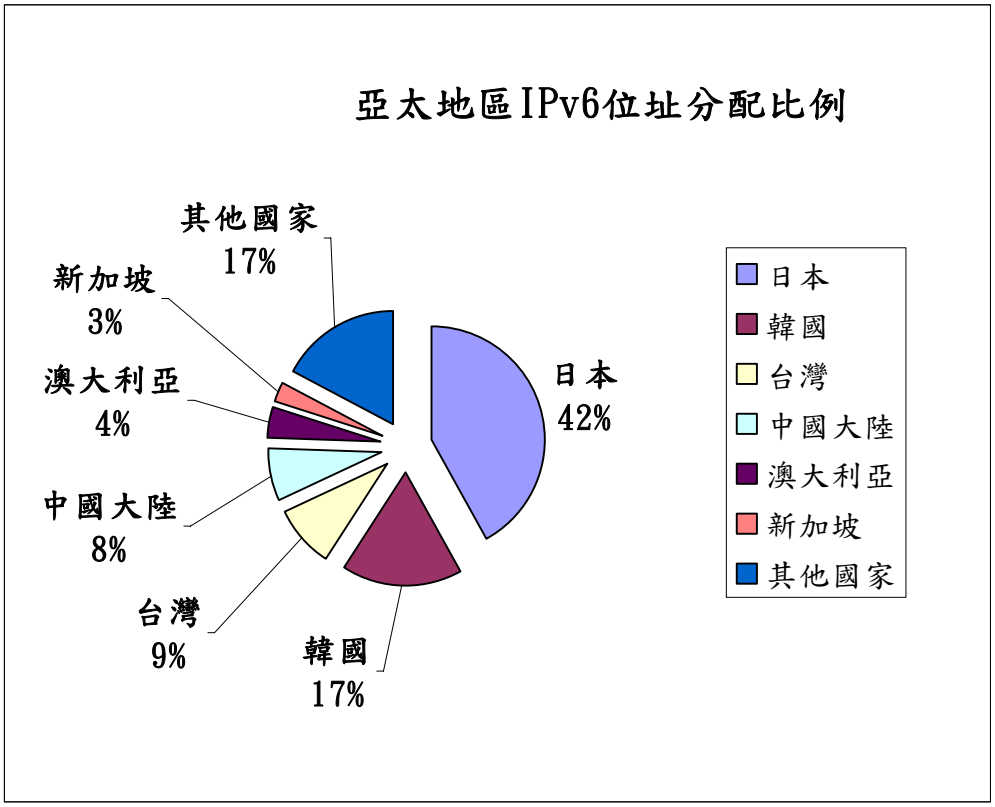


【圖七】 從 IPv4 轉換至 IPv6 之網路運作成本趨勢
（資料來源：IPv6 Promotion Council of Japan）

目前世界網路設備生產大廠，包含 Cisco、3Com、Juniper 都發表多款支援 IPv6 的網路設備，日本與韓國對於 IPv6 的發展也非常積極，日本致力於將既有的資訊家電技術導入 IPv6 功能，使其可於 IPv6 網路中使用；韓國則積極將 HDTV(High Definition Television)升級為 IPv6 的方式傳送；中國大陸也投入大量經費發展 IPv6 相關應用。根據 IPv6 Portal of Taiwan(http://www.ipv6.org.tw)的資料顯示，截至 2004/12/02 為止，亞太地區總共配發 181 個，各國家分配的數量如下【圖八】，數量分配統計圖如下【圖九】所示。其中日本擁有的 IPv6 位址數量最多，其次韓國，台灣則排第三。



【圖八】 亞太地區 IPv6 位址數量分配圖



【圖九】 亞太地區 IPv6 位址分配比例

各國家推廣 IPv6 不遺餘力，台灣主要的 IPv6 計畫是由財團法人台灣網路資訊中心所領導的「我國 IPv6 建置發展計畫」，該計畫是「挑戰 2008：國家發展重點計畫」網路基礎建設「六百萬戶寬頻到家」子計畫項目之一。此計畫為五年計畫，時間從 2003~2007 年，計畫執行內容包含研究發展、標準測試、基礎建設與應用推廣。目前該計畫在 IPv6 骨幹建置、IPv6 關鍵技術研發、IPv6 標準測試…等方面都有多項重大成果。

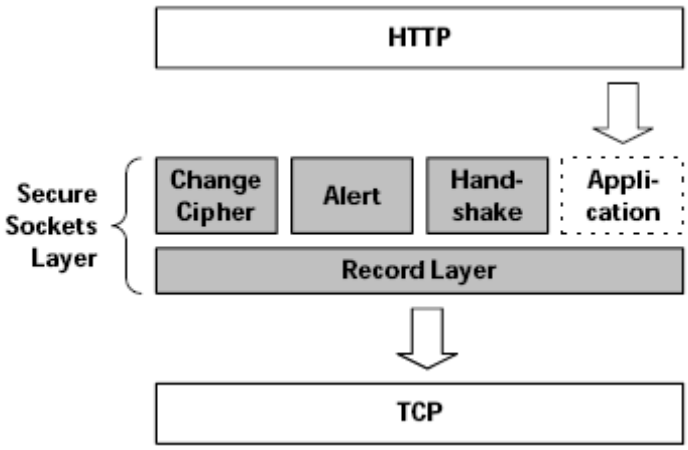
2.3 網頁加密技術

SSL[21]是 Secure Sockets Layer 的縮寫，是一種網路連線資料傳輸加密的協定，透過伺服器憑證加密可以確保網路上其他人不會查閱到網路傳遞的訊息。許多銀行網站也使用 SSL 確保客戶登入網路銀行的帳號與密碼不會被別人取得，該

協定是由網景公司(Netscape)所訂定，目前很多瀏覽器都有支援該協定，使用 SSL 的網址會使用 https://開頭，而不是 http://開頭。

SSL 使用時，伺服器需要申請數位伺服器憑證，申請時需先由伺服器產生憑證要求檔，接著由伺服器憑證認證組織簽章產生伺服器憑證。產生的伺服器憑證需安裝到網路主機上，別人與此主機溝通時，才可使用加密技術做資料加密。

根據如下【圖十】SSL 的架構圖[22]所示，SSL 包含 Change Cipher Protocol、Alert Protocol、Handshake Protocol 與 Application(例如 HTTP)，接著 Record Layer 接收上面四個部分做包裝與編碼後再傳給 TCP 層做傳送。



【圖十】 SSL 的架構圖

2.4 嵌入式系統

目前嵌入式系統已經廣泛運用到大家日常生活之中，許多每天常見的東西，內部其實都是以嵌入式系統為主，小到電子體溫計、血壓計，大到智慧型資訊家

電、掌上型遊戲機、掌上型數位助理器，汽車的控制系統...等等，內部都是以嵌入式系統為主，提供了生活中高科技的應用，並有越來越廣泛應用的趨勢。微軟也看準嵌入式系統的市場，宣布新建 Windows CE 硬體設計中心、微軟嵌入式發展中心[23]。目前最常看到的 PDA 只是嵌入式系統的其中一種應用，將來的 IA 資訊家電，SmartPhone 智慧型手機，也將都是嵌入式系統的天下。

2.4.1 嵌入式作業系統

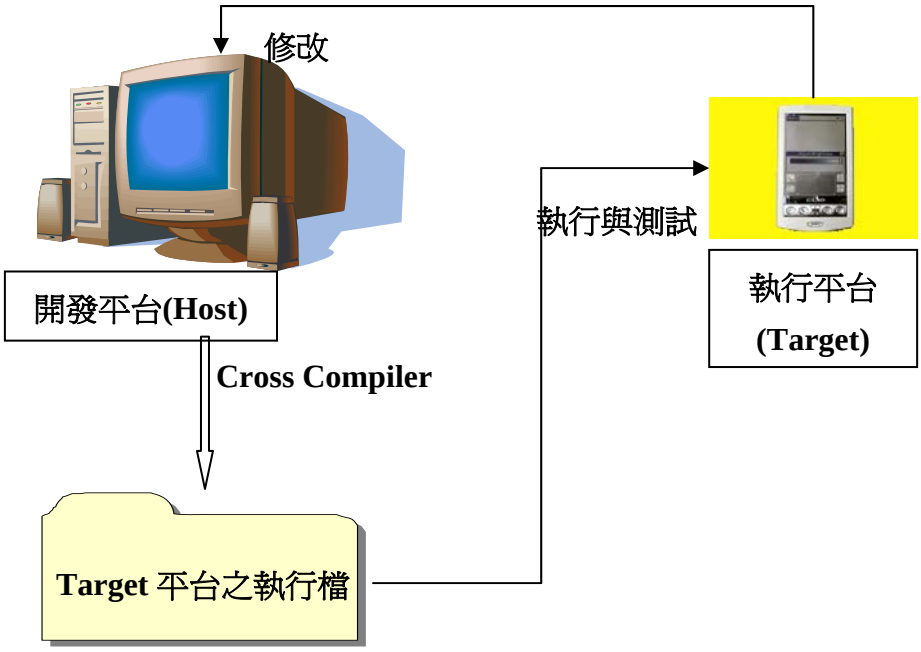
目前嵌入式系統盛行，嵌入式的作業系統也跟著蓬勃發展。嵌入式作業系統與一般作業系統一個簡單的區別方法在於是否有沒有硬碟，一般作業系統儲存於硬碟機中，而嵌入式作業系統都是「嵌入」在可抹寫唯讀記憶體 Flash ROM 中。目前嵌入式作業系統的三大主要使用的為 Palm、WinCE、Linux，三個作業系統比較如下【表五】：

【表五】三大熱門嵌入式作業系統比較表
（資料來源：資策會 MIC，2001/05）

	授權程度/價格	系統大小	所需記憶體	即時性表現	支援硬體	ＡＰ資源	開發工具	多媒體功能	通訊能力
Palm	保守/中	小	小	佳	少	多	標準	普通	普通
WinCE	開放(特定)/高	大	大	普通	多	普通	標準	佳	普通
Linux	開放/免費	小	小	普通	多	多	視廠商而定	佳	視廠商而定

2.4.2 嵌入式系統開發流程

在嵌入式系統的開發流程是非常奇特的，如下【圖十一】所示，奇特的原因在於開發的平台一般為 PC，執行的平台為嵌入式系統，兩者的系統環境並不相同。一般開發時，開發的平台稱為 Host，執行的平台稱為 Target，我們必須在開發平台使用 Cross Compiler 產生出執行平台的執行檔後，把該執行檔直接複製到執行平台中執行，若有問題需回到開發平台做修改後重新產生執行平台的執行檔再測試。經過反覆修改與測試後，最後沒有問題的版本才使用燒錄的方式寫入到執行平台的 ROM 或 Flash 中，如此才完成嵌入式環境程式的開發。



【圖十一】 嵌入式系統開發示意圖