第三章 潛在的問題與改進構想

3.1 服務搜尋協定整合潛在問題

從第二章的例子，我們可以發現不同的服務搜尋協定的特性，也可以發現不同的服務搜尋協定功能上不同的地方，如表 3.1 所示。

表 3.1 SLP、UPnP、Jini、Bluetooth SDP 功能比較表

特性	Jini	UPnP	SLP	Bluetooth SDP
架構	Client-Server / Peer-to-Peer	Peer-to-Peer	Client-Server/ Peer-to-Peer	Peer-to-Peer
目錄服務	Lookup Service	No	Directory Agent	No
Lease 機制	Yes	Yes	Yes	No
屬性搜尋種類	Service type Service ID Attributes	Device type Service type	Service type Attributes String	Service type Attributes
遠端呼叫	RMI (Service Proxy Object)	SOAP	未定(SOCKET)	未定 (RFCOMM/OBX)
訊息格式	Object Serialize	XML	ABNF	Stream
Multicast 機制	Reg:224.0.1.85:4160 Ann:224.0.1.84:4160	239.255.255.250:1900	239.255.255.253:427	Inquiry scan
Advertise 機制	Lookup Service	Multicast	SLP DA/Daemon	SDP
支援型別	支援複雜資料型別	Integer/Float/String/ Boolean/Date/ Base64/Hex/UUID	String/integer/ Boolean/keyword/ opaque	Long/String/ Calendar/byte[]
優點	1.功能最強大，服務可執行於客戶端或是呼叫 RMI stub 2.提供 JavaSpace、Transaction 機制	1.不限語言平台 2.提供基本型別的遠端控制 3.具 presentation 規範和 AutoIP 機制	提供豐富的服務屬性查詢機制	可於 Mobile Device 上使用
缺點	1.需要 JVM 2.硬體需求較高	1.功能單純 2.每個設備需提供 Web Server 功能	1.僅規範服務搜尋的方法，不具遠端呼叫機制 2.需 DA 或 SLP Daemon	僅規範搜尋服務機制

我們就不同特性，分別說明設計家用閘道服務時，會遇到的問題，並提供解決方法來進行整合服務：

1. 集中式或非集中式架構：

服務搜尋機制主要可以分為 2 種類型，一種是具備集中式的目錄服務，所有的服務資訊都會記錄在目錄服務裡 ，像 Jini 的 Lookup Server 和 SLP 的 DA。客戶端會使用 Client/Server 的機制和目錄服務溝通取得服務資訊。另一種是非集中的方式，服務會利用廣播(Advertise)的方式通知其它成員；當客戶端要搜尋服務時，利用 Multicast 向各個服務詢問，這種是屬於 Peer-to-Peer 的方式，像

UPnP 和 Bluetooth SDP 就是採這種方式。而 SLP 當不具備 DA 時亦是這種方式。Jini 則是當在進行尋找 Lookup Server 時，也是採行 Peer-to-Peer 的方式。如圖 3.1 所示。

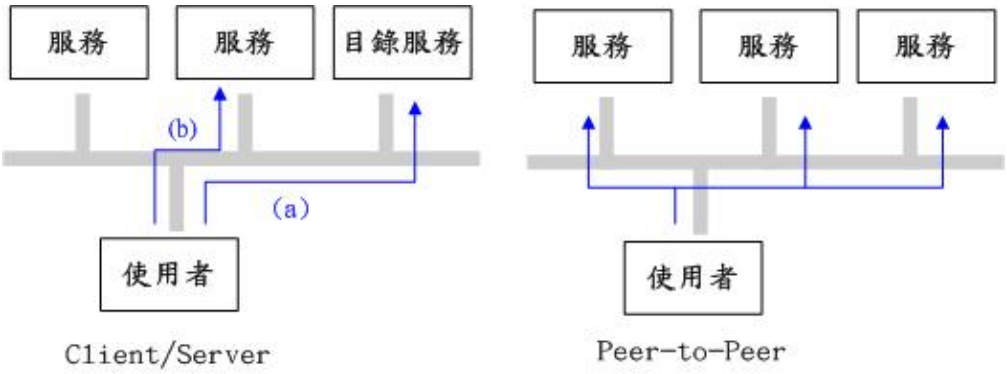

圖 3.1 服務搜尋機制架構

在家用閘道整合下，閘道器需知道不同的服務搜尋網路下有提供那些服務資訊，因此，在 Client/Server 的架構上，閘道服務需週期向目錄服務詢問，以維持最新的線上服務資訊，而在 Peer-to-Peer 的架構上時，則需使用 Listener 來聆聽服務的 advertise 訊息，並週期的詢問網路上仍還在線上的服務，以避免服務已離開，但未送出離開訊息。

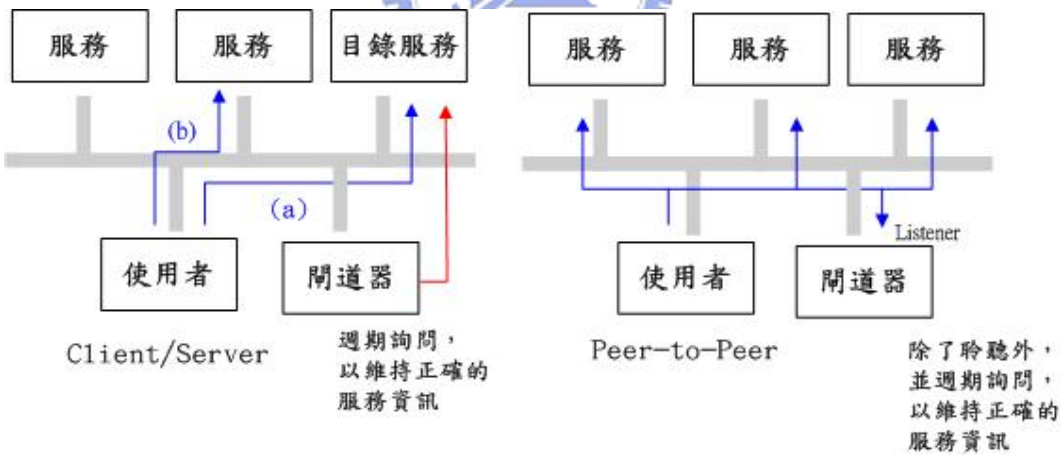

圖 3.2 閘道服務在不同架構上的處理方式

2. 服務屬性搜尋種類：

不同的服務搜尋機制提供不同的服務屬性查詢機制；使用者可藉由服務屬性來詢問是否有符合自己需要的服務。可查詢的服務屬性包括服務類型、服務 ID、自訂屬性、設備類型和字串等。亦即當服務被 Wrapper 至另一 Domain 時，它的屬性資料要一併複製過去，在這由閘道器處理一般性質的屬性轉換，而忽略特殊自訂型別的屬性。如圖 3.3。

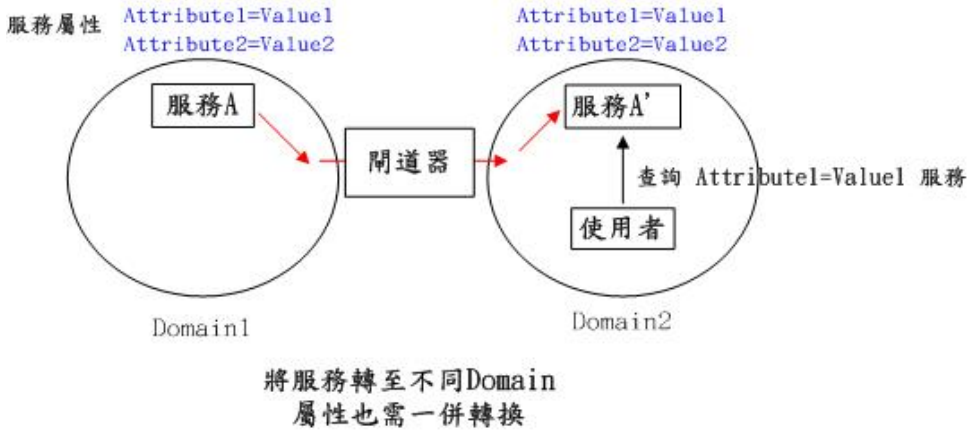

圖 3.3 閘道服務進行服務屬性轉換

3. 遠端呼叫方式：

SLP 和 Bluetooth SDP 僅提供服務搜尋協定，並未定義遠端呼叫和資料交換的協定，因此，在系統模擬時，會先由系統定義 SOCKET 方式的資料交換的協定。Jini 使用 RMI 來做遠端呼叫，而 UPnP 則採用 SOAP 來進行遠端呼叫。RMI 客戶端是以 Stub 和 Skeleton 來和伺服端進行溝通，而 SOAP 則利用 SOAP 封包封裝呼叫資訊，和伺服端進行溝通。如圖 3.4。

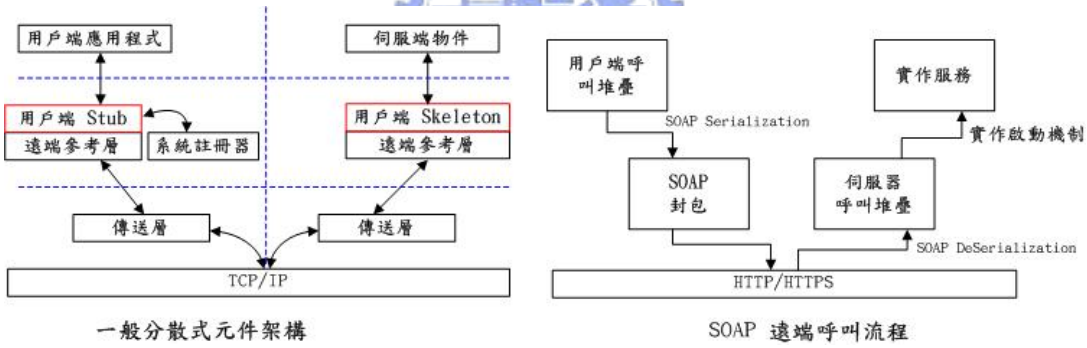

圖 3.4 一般分散式元件運作架構和 SOAP 遠端呼叫流程

但當服務搜尋機制是使用分散式元件做遠端呼叫機制時，便會發生需動態產生 Stub/Skeleton 來遠行遠端呼叫的問題。如圖 3.5。

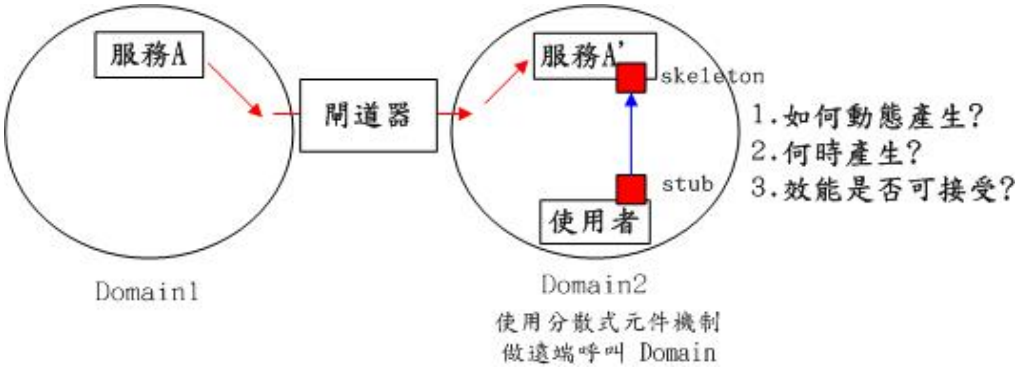

圖 3.5 服務搜尋機制 Stub/Skeleton 動態產生情況

當服務轉至使用分散式元件機制做遠端呼叫的 Domain 時，服務需動態產生 Stub/Skeleton 問題，主要有三種解決方式：

(1)由閘道器線上動態產生 Stub/Skeleton 程式碼，並進行編譯、部署安裝。

優點：最容易達成。如圖 3.6。

缺點：閘道器負擔太重，會變成效率瓶頸所在。

(2)使用 bytecode 產生器（例如 Javassist）動態產生 Stub/Skeleton，跳過編譯動作，直接部署安裝。如圖 3.7。

優點：跳過編譯動作，速度較快些。

缺點：仍需要進行部署，閘道器負擔仍重。

(3) 不產生新的 Stub/Skeleton，利用閘道器提供服務傳送呼叫資訊給客戶端，由客戶端經由閘道器服務回應要呼叫的服務，由 閘道器利用 Reflective，於閘道器端即時代理呼叫 。如圖 3.8。

優點：閘道器轉換服務至另一 Domain 時，毋需處理 Stub/Skeleton 動態產生的問題。

缺點：閘道器需提供傳送呼叫資訊和代理呼叫的功能。

圖 3.6 線上動態產生 Stub/Skeleton

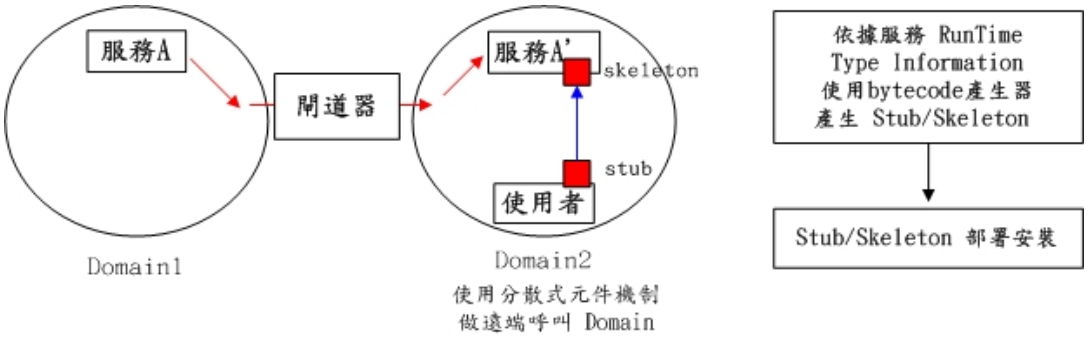

圖3.7使用bytecode產生器動態產生Stub/Skeleton

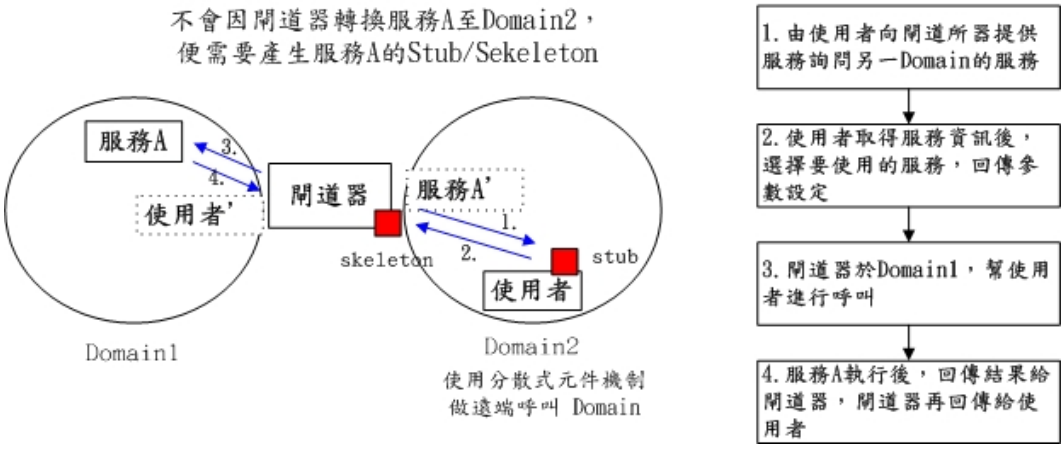

圖 3.8 使用閘道服務傳送服務資訊

考量閘道器的效率和單純性，採用方法三、使用閘道服務傳送服務資訊方式來解決分散式元件 Stub/Skeleton 的問題。

4. 訊息資料格式：

家用閘道服務會跨不同服務搜尋機制的網路，因此至少需連接 2 端不同的網路，需具備處理不同網路訊息的能力。如圖 3.9。

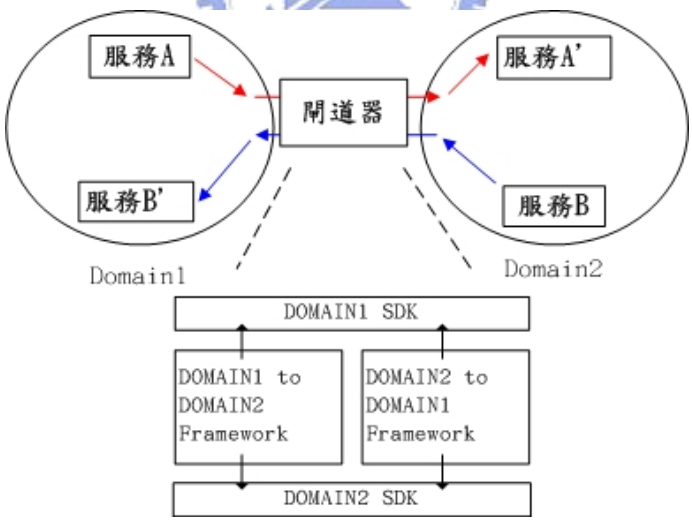

圖 3.9 閘道器內部的 Framework 設計

5. Multicast/ Advertise 機制：

不同的服務搜尋機制，藉由閘道器內部的 SDK 來處理個別的機制。

6. 支援的資料型別：

Jini 支援複雜資料型別，由於在不同服務搜尋機制下支援的資料型別個不相同，當閘道器轉換服務時，也需進行資料型別的對應。因此，在此僅支援共

同的資料型別，如：Integer、String、Boolean、Float，若是屬於複雜型別或自定型別的參數型態，則該 method 便不加以轉換。有共同的資料型別轉換機制，不同的服務才可相互溝通。如圖 3.10。

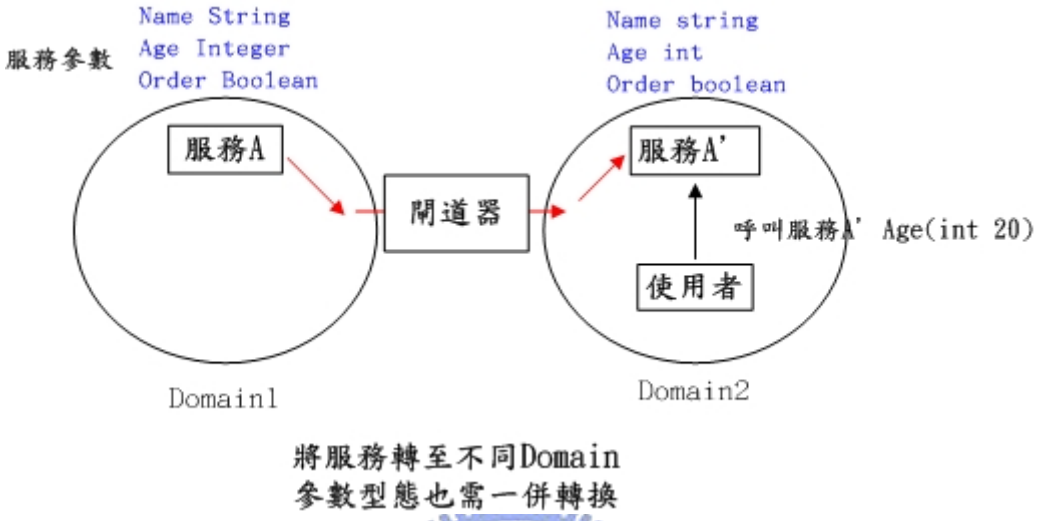

圖 3.10 閘道器處理型別轉換機制

最後，列出上述不同服務搜尋機制整合會面臨的問題，和解決方法，如表 3.2 所述。

表 3.2 不同服務搜尋機制下，閘道整合解決方案

問題	解決方法
架構不同	Client-Server ：閘道器週期詢問，以維持正確的服務資訊 Peer-to-Peer：閘道器除了聆聽外，並週期詢問，以維持正確的服務資訊
服務屬性搜尋種類不同	閘道器處理一般性質的屬性轉換
遠端呼叫方式 Stub/Skeleton 問題	使用閘道服務傳送服務資訊方式來解決
訊息資料格式不同	閘道器需具備處理不同網路訊息的能力
Multicast/Advertise 機制	由閘道器內部的 SDK 來處理個別的機制
支援型別	僅支援共同的資料型別，如：Integer、String、Boolean、Float 對應轉型

3.2 網路閘道架構概觀

網路閘道的設計方法主要可分為二種：Service query translation 和 Service registration translation 方法[2]，圖 3.11 描述這二種方法。

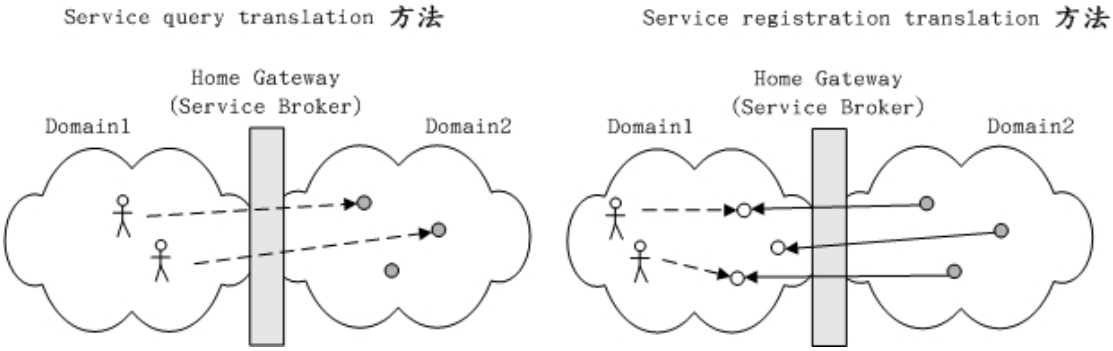

圖 3.11 網路閘道設計方法

- Service query translation：將使用者的 Query 經由轉換，送至其它的 Domain，服務執行後的結果，再轉換回原 Domain。而服務的 Service Registration 則不轉換至其它 Domain。因此，只有 Query 和 Result 會在經過閘道器時進行轉換，轉換至適當的 Domain。另外，除了 Query 需轉換外，Notification 也需經由閘道器進行適當轉換。
- Service registration translation：將 Service Registration 送至其它需要的 Domain，經由閘道器進行轉換至符合該 Domain 的 Service Registration，而使用者的 Query 直接跟 Registration 溝通，不再經過轉換。

這二個方法主要的不同在於 Service query translation 是利用閘道器來轉換換使用者的 Query request 至不同的 Domain，而 Service registration translation 則是將 Service Registration 服務轉換至不同的 Domain。在這選擇採用 Service registration translation 方法來實作，主要的原因如下：

- 每個 Service Discovery 機制所提供 Query 和 Notification 的功能不同，若由單一 Domain Query 轉換至其它 Domain，會造成 1:M 的轉換，每一個 Domain 閘道的轉換服務皆需具備 M 種功能。因此，直接使用 Registration 轉換至各 Domain，直接面對單一 Domain Query，是較佳的方法。
- Service Discovery 機制的執行需依賴於靜態的 Service Registration 進行操作。因此 Service Query 較 Service Registration 常改變和傳送。如果，採行 Service query translation 的方法，閘道服務可能會因為使用者不斷的查詢而變成一個效能瓶頸所在，所以，本研究採行 Service registration translation 方法。

- Service Discovery 的基本操作包括採用集中式或分散式儲存 Service Registration 服務、更新服務 Attribute 和查詢服務，也就是說需要同步(Synchronize)維護分散在不同 Domain 的 Service Registration 狀態。 如果考慮同步問題，那使用 Service registration translation 較易達成。

由於閘道服務必需在不同的 Domain 間進行適當的 Registration 轉換。而且需連接兩個以上不同的 Service Discovery Domain，因此，不可採行單一 Domain 對映至其它不同 Domain，這樣會變成 Quadratic growth，而應該各個 Domain 對映至單一的 Generic Domain。如圖 3.12 描述。

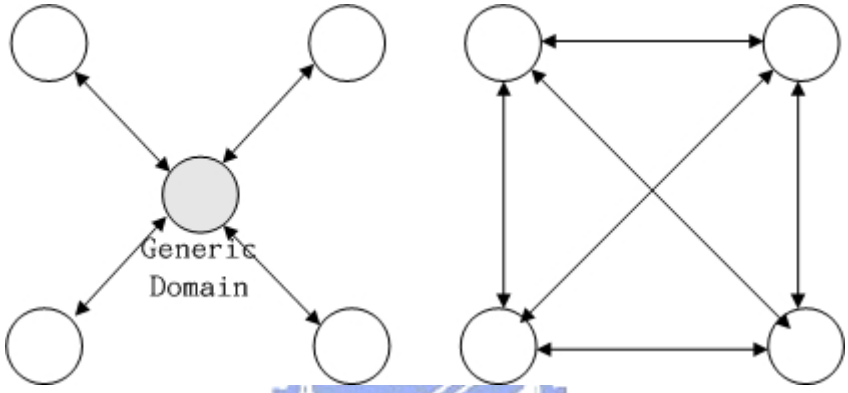

圖 3.12 Generic Domain 和不具 Generic Domain 網路架構

各個 Domain 閘道服務負責處理外部 Generic Domain 和內部 Domain 間 Service Registration 的轉換。在本篇架構裡，決定使用 Jini Domain 做為 Generic Domain，一方面是由於 Jini 的功能強大可以對映至其它 Domain，另一方面是使用 Jini 做 Generic Domain 也可減少 Jini Domain 的服務便不需經由閘道服務，可直接其它 Domain 溝通。如圖 3.13。

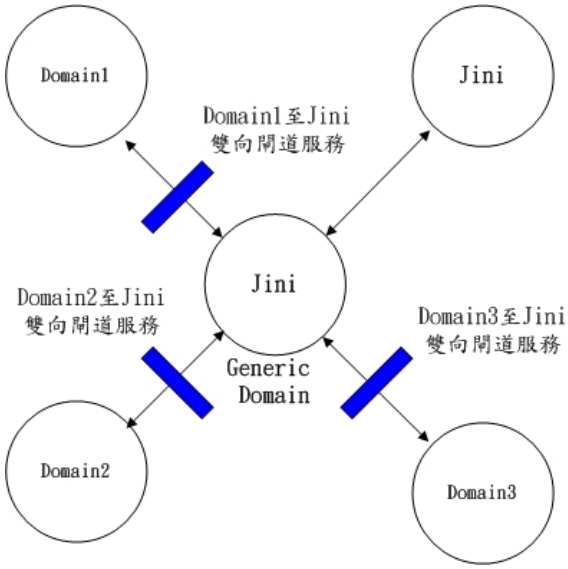

圖 3.13 使用 Generic Domain 和閘道服務連接不同 Domain