# 基於 CoAP 且適用於受限環境的智慧家庭服務發現架構
# Constrained Resource-Oriented Service Administration (CROSA)


胥沛恩[1]、廖峻鋒[2]、林明言[1]
逢甲大學資訊工程系[1]，國立政治大學資訊科學系[2]
Pei-En Hsu[1], Chun-Feng Liao[2], Ming-Yen Lin[1]
Department of Information Engineering & Computer Science
Feng Chia University[1]
Department of Computer Science
National Chengchi University[2]
Email: m0303799@mail.fcu.edu.tw, cfliao@nccu.edu.tw, linmy@mail.fcu.edu.tw



**摘　要**

智慧家庭應用在現今生活當中，越來越受到各家廠商的青睞，因此其設備與設備之間，相互通訊以及相互應用的效能也越來越重要；傳統服務發現協定往往未考量在智慧家庭中，資源較受限的設備，因此許多服務發現協定對於受限設備並不合適。基於上述原因，本研究將以 CoAP 與 MQTT 兩大物聯網通訊協定為基礎，設計一套具備 Web of Things 概念，且適用於受限設備的輕量級資源導向智慧家庭服務架構；其中，本論文將會以 CoAP 為基礎實作架構底層的服務發現協定：CROSA，再以 MQTT 延伸作為架構中應用層的傳輸方法，以達到既符合受限設備使用環境要求，更確保在套用在智慧家庭中應用時，能有更直觀且靈活的佈局方法。



**關鍵字：**智慧家庭、物聯網、服務發現、MQTT、CoAP


## 一、前言

物聯網(Internet of Things，IoT)是近年來重要的研究議題，其核心概念是將所有物品透過 Internet 連結，使每個物體都能被定址且相互通訊。在智慧家庭的應用中的核心概念也與物聯網極為相似，在一個智慧家庭系統中，每個家電或設備可以視為一個節點，每個節點可以提供一個或一個以上的服務，且這些節點所提供的服務往往是其他節點所需要的。因此，在智慧家庭中每個節點都需要即時的瞭解與知道現在網域內有哪些服務存在，且如何使用特定服務或如何操控特定設備。

在近幾年，由 IETF 的 CoRE(Constrained RESTful Environment) 工作組所制定的 REST(Representational State Transfer)模式的通訊協議 CoAP (Constrained Application Protocol)替受限設備提供一種較為經濟、輕量且方便的通訊方式；CoAP 是基於 IP 網路的方式，以 URI 替每個終端設備所提供的資源或服務定址，也因為如此，外部設備可透過特定 URI 取得所需服務與資源。CoAP 替受限設備的開銷與定址問題找到了良好的解決方案，但 CoAP 所採用的通訊模式卻與智慧家庭習慣採用的 publish/ subscribe 模式大相逕庭，CoAP 採用的 Server/Client(Request－Response)架構，需要 Client 端主動向 Server 端取得所需資訊，在智慧家庭空間中，單資服務所提供的資源很多時候都同時必須要提供給許多節點使用，也因此，採用 Server/Client(Request－Response)的模式佈局智慧家庭空間，在實務上並不適合。因此，本論文將結合 MQTT(Message Queuing Telemetry Transport)做為整體架構的應用通訊層，MQTT 的傳輸模組以 publish/ subscribe 的模式進行訊息交換，在事件通知的實現上較 CoAP 更容易；這樣的分層架構，可以使在智慧家庭應用中，服務發現與訊息交換的工作更具靈活與彈性。


* 本論文部份研究受科技部研究計畫經費補助，編號: 104-2221-E-004-001、104-2627-E-002-006、與 104-3115-E-004-001。


## 二、相關研究

CoAP (Constrained Application Protocol)是由 IETF 的 CoRE(Constrained RESTful Environment)工作組所制定的 REST(Representational State Transfer)模式的通訊協議，替受限設備提供一種較為經濟且方便的通訊方式，CoAP 的傳輸層是使用 UDP 做為傳輸方式，相較於 HTTP 使用 TCP 做為傳輸層，且 CoAP 在封包 header 上所需的長度較短、以及降低封包在分析時的複雜性，因此 CoAP 更符合受限設備的應用。CoAP 也包含了以下功能：(1)單點傳播 (uincast) 與群播 (multicast) 、(2) 告知收到 (acknowledged)與不用告知收到(unacknowledged)的傳輸方式、(3) 與 HTTP 相似的四個不同請求的方法：GET(檢索資源), POST(處理請求), PUT (更新/創建資源) 與 DELETE(刪除資源)、(4) 以及三個不同類型的回應碼(response codes)：2.xx (success), 4.xx (client error), 5.xx (server error)。但 CoAP 通訊協定所使用的 Server/Client 模式，相較於 MQTT 的 publish/subscribe 方法，對於智慧家庭在應用層開發的需求上，MQTT 的 publish/subscribe 的方法更直觀、也更符合智慧家庭應用當中，應用層傳輸上的概念。

在智慧家庭服務發現協定相關的研究當中，在文獻[10]的研究當中，提除了一個基於 REST 架構風格來改良 UPnP 的服務發現管理方式 ROSA(Resource-Oriented Service Administration)，ROSA 藉 UPnP 操作風格所設計，以 REST 軟體風格取代 SOAP，藉此改善效能；但 ROSA 並未針對

受限設備做設計，且 ROSA 在應用操作仍採用 Server/Client(Request－Response)的模式，在智慧家庭空間中，單資服務所提供的資源很多時候都同時必須要提供給許多節點使用，也因此，採用 Server/Client(Request－Response)的模式佈局智慧家庭空間，在實務上並不這麼適合。

另外關於 CoAP 的相關研究當中，多以大部分都是有關於如何整合或者擴展 CoAP 本身的核心功能為主：在文獻[13]中，提出了一個基於 CoAP 整合資源供受限設備使用的架構，該架構透過建立 Entities 來整合多個實體資原，並透過 Entities 實作服務發現與管理，Entities 以 CoAP 為基礎實作，供 CoAP Client 使用，但該架構過度依賴 Entities，且缺乏替代機制，若 CoAP Client 所使用的 Entities 發生不可預期的問題而中止服務時，則整個架構就會停擺運作；然而本研究的服務發現方法是採分散式管理機制，不會因為單一目錄遺失或發生問題而停擺。在文獻[14]這篇研究當中，透過建立一個 Proxy 接受來自 HTTP Network 的請求，Proxy 會針對該請求欲找尋的服務與資源，透過 multicast 的方式在 CoAP Network 底下對多個 CoAP Server 做服務發現，並整合多個來自 CoAP Sever 的回應給請求的 HTTP Client；在這樣的架構底下，有著與文獻[13]相同的問題，太過依賴 Proxy 所提供的功能，使得若 Proxy 發生問題，則整個架構會因此停擺不能使用；加上文獻[14]中的架構仍採用 Server/Client(Request－Response)的模式，因此在智慧家庭空間的應用上比起本研究所採用的 publish/subscribe 方法更顯不適合。

目前關於 CoAP 與服務發現相關研究當中，鮮少採用分層架構去做設計，因此往往會因為採用單一協定實作服務發現功能時，會因協定本身的缺陷而有所限制。本研究所提出的架構與目前大多數所提出的方法最大的差異在於，我們透過分層式架構，彌補了 CoAP 本身所帶來的限制，進而將 CoAP 的優點最大化的運用在服務發現的功能當中。

## 三、實作方法

### 3.1 CROSA 的系統架構

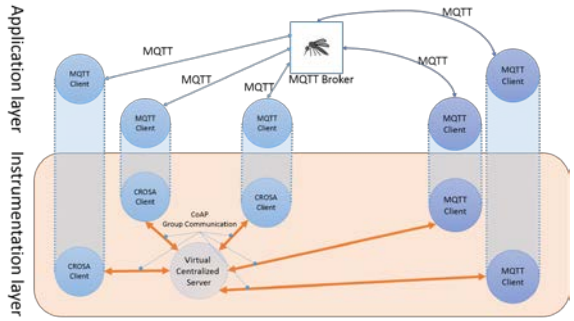


圖 1 CROSA 系統架構與分層

Constrained Resource-Oriented Service Administration(CROSA)，為一個以 CoAP 為基礎且 MQTT 為應用層傳輸方式的服務發現管理協定(如圖 1)。CROSA 結合 CoAP Group Communication 相關概念與技術，實作一個適用於受限設備與受限網路環境的服務發現協定。CROSA 為一個不需中介目錄的服務發現協定，且符合 REST 規範，讓設備在進行服務發現等相關功能時，有統一的介面去實作與使用。

在 CROSA 的架構中，有四層面相的考量：(1) Discovery 、 (2)Description 、 (3)Control 以及 (4)Eventing。其中，我們在 Discovery 與 Description 是以 CoAP 為基礎的方法去實現；然而，在 Control 與 Eventing 中，我們則是將實際的工作交由 MQTT 來執行，會這麼做的原因在於採用 publish/subscribe 通訊模式的 MQTT，比起採用 Server/Client(Request–response)模式的 CoAP，在智慧家庭中應用上更為方便且經濟。

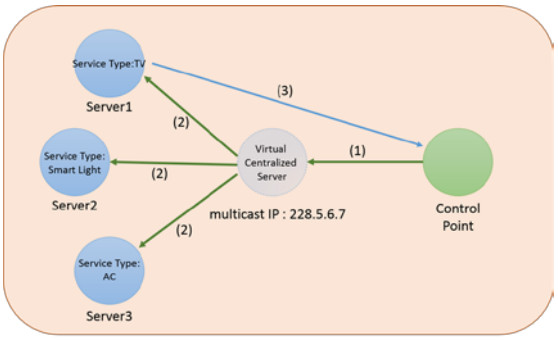


圖 2 CROSA 服務發現架構示意圖

CROSA 以 IP Multicast 方式實作相關功能，我們透過圖 2 來說明 CROSA 的架構。在 CROSA 架構中(如圖 2)，有一個 IP 群播位置 228.5.6.7，我們可以將它視為一個 Virtual Centralized Server，也因此，我們也可以將 CROSA 節點透過 IP Multicast 所發出的 REST 請求是向該 Virtual Centralized Server；透過這樣的概念，我們可以很輕易地對映到在一般我們理解的 Client 端與 Server 端的請求與回應的角色：當一個 Control Point 要透過 CROSA 取得一個服務時，如：TV；則 Control Point 會向這個 Virtual Centralized Server 發出一個 GET Request(假設：GET /.well-known/core?st=TV)，如圖 2 中(1)；對於 Control Point 而言，其請求對象只有一個，就是 Virtual Centralized Server，但透過 Virtual Centralized Server 則會轉發該請求至所有在 IP Multicast Group 內的成員，如圖 2 中(2)；當 Group 下所有成員都收到由 Control Point 所發送請求後，會分析其請求是否符合其服務類型(Service Type)，若符合，該 Server 會直接回應 Control Point 服務的 Description，如圖 2 中(3)；Description 為描述服務提供的資源與如何使用的方法，在後面小節會詳述 Description。在圖 2 的例子當中，我們可以瞭解 CROSA 訊息交換的原理與原則，且基於以下裡由，因此我們使用 Virtual Centralized Server(也就是分散式服務發現協定)設計我們的服務管理機制：

- 強健性(Robustness)高：不需要一個實體的中

介目錄，且 Virtual Centralized Server 實際上是不存在的，因此不需擔心使用中介目錄時，目錄斷線、發生異常時，服務發現機制無法正常運行。

- 布署成本較低：因為不需要額外準備中介目錄所需的硬體設備，因此成本較低，且一般來說，中介目錄因需要管理所有服務、分析、回應請求，因此其硬體效能需要較高，故所需成本也較高。
- 較低開銷、較高效能：在文獻[11]中提到，服務發現協定中，使用分散式比起集中式，擁有較低的訊息傳輸量與較高的效能；其原因在於集中式管理的服務發現協定還要需向目錄做完整的註冊動作，且目錄也需要透過輪詢或者其它主動機制隨時更新目錄的狀態，以保持目錄最新狀態；因此，集中式需要有較多的訊息交換，以致網域中可能存在大量的驗證封包。因此，在有限網路與設備的應用情況下，使用分散式服務發現協定較適合。

在後面的小節當中，我們將針對智慧家庭當中需考量的面向：Discovering Things、Controlling Things、Notifications from Things 及 Representing Things；也就是文獻[10]中，ROSA 的四個面向：Discovery、Description、Control 以及 Eventing，說明 CROSA 設計的原理與運作方式。

### 3.2 CROSA 發現 Discovery

在 CROSA 的發現過程中，我們採用 CoAP Group Communication 相關技術實作發現機制，CoAP Group Communication 是基於 IP Multicast 方式來實作[12]。在 CROSA 中，因為是採用分散式服務發現的設計方案，因此 CROSA 不須仰賴集中式的目錄，也因而降低 CROSA 中，節點與目錄的耦合性，節點不會因為目錄的遺失而無法運作。在 CROSA 可以循兩種模式發現可用服務：(1)Directory Search、(2) Local Search。

Directory Search 是透過自身暫存的廣告訊息(advertisement messages)直接取得服務。當有服務加入到由 CROSA 所管理的受限網路時，會向 Virtual Centralized Server 發送 PUT 訊息，Virtual Centralized Server 會轉發該請求至 Group 內的其他成員有新的服務加入，加入的新服務會以 PUT 訊息發送至以下 URI：

coap://228.5.6.7/.well-known/newService

這是 CROSA 協定中公眾的 URI；其中，我們可以在這個 URI 中加入參數(payload parameter)，來告知自己是屬於哪種 Service Type(假設為 TV)，因此在新服務加入 PUT 操作的 URI 可以表示成：

coap://228.5.6.7/.well-known/newService?st=TV

如此一來，透過 Virtual Centralized Server 接收到加入訊息的成員，就可以知道新加入的這個服務是屬於哪種類型的服務(Service Type)，並依照自身興趣選擇性加入這個新的服務進節點本身的目錄中。詳細過程如圖 3。

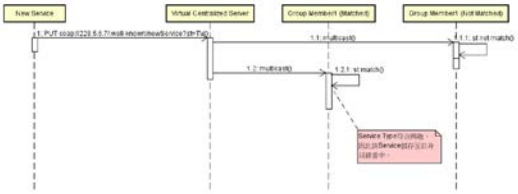

圖 3 新服務加入 CROSA 與感興趣成員加入新服務至目錄流程

舊成員(也就是比新服務先加入 CROSA 的節點)，會依照它們現在或未來可能會使用到的服務類型選擇性加入新服務至目錄，也因此，當舊成員有要使用某項服務時，會先搜尋自身目錄，若有興趣的 Service Type 時，就會向該 Service 發送 GET 訊息，取得該服務的 Description，而這個 GET 訊息則是直接與 Service 所在的 Server 做操作，而取得 Description 的 URL 位置為/.well-known/profile，如圖 4；因此假設該 Service 的位置為 192.168.1.1 則，其 GET 的 URI 如下：

coap://192.168.1.1/.well-known/profile

當 Control Point(舊成員)透過這樣的機制取得感興趣 Service 的 Description 後，就可以做後續的處理。這樣的機制，省去了透過 IP Multicast 尋找感興趣服務的過程，因此透過這個模式找到服務的效率最快。

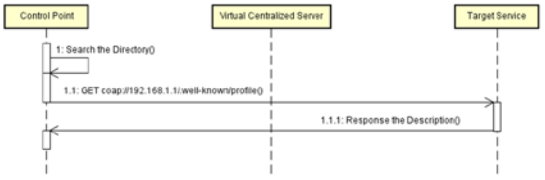


圖 4 Directory Search 流程圖

當然，有些時候，Control Point 要找尋感興趣服務時，自身目錄可能沒有當前需要的服務類型，亦或者目錄中的服務端當前不在線上或者其它因素無法使用時，就必須仰賴另一種模式。

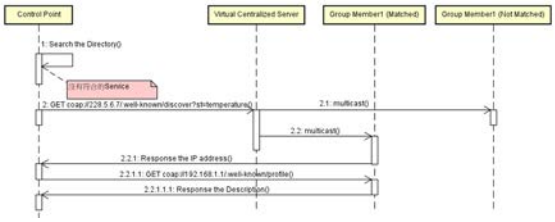


圖 5 Local Search 流程

Local Search 是向 Virtual Centralized Server 發送 Discovery 請求尋求可用服務。當 Directory Search 模式無法找到所需的服務時，就必須透過 IP Multicast 方式來搜尋可用服務。在前面圖 2 的範例當中，我們已經略窺一二 Local Search 的運作方法，在這邊我們把 Local Search 做更詳細的敘述。

舉例來說，當一個 Control Point 在自身目錄找不到可用服務時，會向 Virtual Centralized Server 發送 GET 請求，該請求內包含欲請求服務類型(Service Type)，假如 Control Point 要找尋溫度感測

器(st=temperature)，則 Control Point 會透過事先規定好的 URI：/.well-known/discover 並附上參數(payload parameter)來搜尋可用服務，其 URI 如下：

coap://228.5.6.7/.well-known/discover?st=temperature

若網域內有符合 st=temperature 的服務時，該服務會回應其 IP 位址，而 Control Point 則會透過該 IP 位址直接與該 Service 所在的 Server 請求該服務的 Description，請求該服務的 Description 方法一樣是透過 GET 請求，發送至該服務的 /.well-known/profile 來取得，如果該服務的 IP 為 192.168.1.1 時，其 URI 如下：

coap://192.168.1.1/.well-known/profile

綜合上述，Local Search 流程可以以圖 5 來表示。

在 CROSA 的 Discovery 機制中，Control Point 先透過 Directory Search 在自身目錄內服務是否有符合且可用的 Service 存在，若有則直接向該 Service 所在的 Server 發送 GET(/.well-known/profile)請求取得 Description。倘若無法透過 Directory Search 找到符合且可用的 Service 時，Control Point 則會透過 Local Search，向 Virtual Centralized Server 發送 Discovery 請求尋求可用服務。

### 3.3 CROSA 描述 Description

傳統服務發現協定，服務的 Description 許多都是以 XML 語言來描述，如 UPnP 以及文獻[10]所提出的 ROSA 發現協定，都是以 XML 語言來做服務描述，雖然 ROSA 以 WADL 取代 UPnP 的 SOAP 協議，但對於受限設備而言，解析 XML 是一件非常消耗資源與記憶體的事情，這也是為什麼這類型的服務發現協定無法適用於受限設備的原因。

表 1 CROSA 服務描述檔範例

```
{"deviceInfo":{
        "deviceName":"Temperature Sensor",
        "deviceMQTTid":"tempSensor01",
        "serviceType":"tempSensor",
        "ipAddress":"192.168.1.1"
        "port":"5683"},
    "resources":{
        "base":"http://localhost/",
        "resource":{
            "path":"/tempCelsius",
            "method":{
                [{"name":"GET",
                      "request":{},
                 "response":{
                     [{"mediaType":"application/json"}]
                 }},
                 {"name":"PUT","request":{
      ["name":"request","style":"plain","type":"xs:int"]
                      },"response":{
                     "mediaType":"application/json"
                 }
                 }
               ]}
        }
}
```

在 CROSA 則是以較輕量的 JSON 格式取代 XML 做為服務的描述方法(如表 1)，JSON 比起 XML 有更快的解析速度、占用更少的記憶體空間，因此 CROSA 用 JSON 代替 ROSA 所使用的 WADL 來描述服務與設備。

### 3.4 CROSA 控制 Control

設備在 CROSA 服務發現協定這個 layer 當中，會將實際的 Control 交由 MQTT 去執行，CROSA 僅僅替 Control Point 與目標服務提供 MQTT 訂閱與發佈所需的相關資訊，因此 CROSA 並不會對目標服務進行實際操控，也因為如此，CROSA 可以更專心地執行服務管理機制，其餘應用層相關操作，則交由 MQTT 去執行。

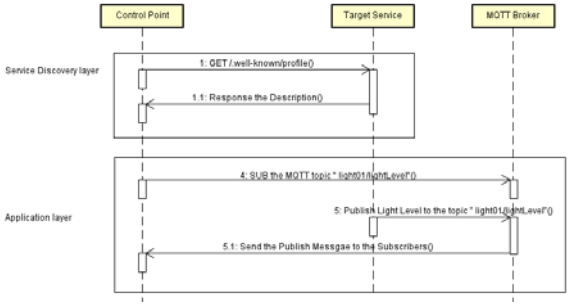


圖 6 CROSA Control 機制中取得(GET)服務狀態流程

在上一個小節 CROSA 的描述範例(表 1)當中可以看到，當設備透過 GET 請求取得服務以後，服務端會透過 CROSA 回應設備”mqttTopic”這個值，其目的就是在於提供整體架構中 Application layer 所需的資訊。我們舉個例子：如果某個 Control Point 要取得一個智能燈的狀態時(如圖 6)，Control Point 會透過 CROSA 服務發現協定取得該服務的 Description，再透過 Description 取得該服務在 Application layer(也就是 MQTT)所在的 MQTT Topic 為何，在取得服務所在的 MQTT Topic 名稱以後 Control Point 就會透過 Application layer 訂閱該服務的 MQTT Topic，以取得所需資訊。

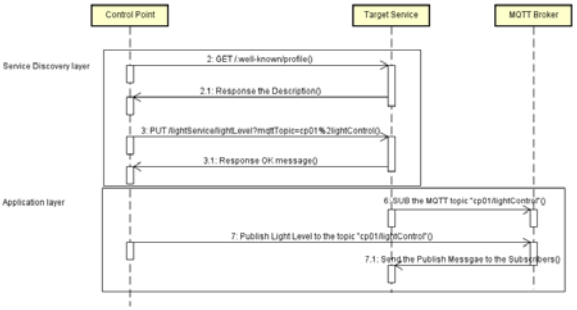


圖 7 CROSA Control 機制中更動(PUT)目標服務狀態的流程

然而，如果要對目標服務做資源得更動時，繼續以智能燈為例(如圖 7)，如果 Control Point 要控制目標服務(也就是智能燈的亮度)時，Control Point 會透過 CROSA 取得智能燈的 Description，再透過 Description 找到對應的 PUT 方法，並附帶 Control

Point 的 MQTT Topic 的參數(payload parameter)給智能燈，在智能燈收到 PUT 方法後，會透過 Application layer(也就是 MQTT)訂閱 Control Point 所指定的 Topic 以接收 Control Point 所要更動的資源狀態。

當 Control Point 透過 CROSA 的 Control 機制，取得(GET)目標資源的狀態後，因為在 Application layer 已經訂閱了該資源狀態的 Topic，因此，往後若目標資源狀態有所變更時，Control Point 不需要額外進行 REST 中 POST(訂閱)的操作，即可透過 MUG 立刻知道目標資源的狀態變更。另一方面，Control Point 透過 CROSA 的 Control 機制，更新過目標資源以後，因為目標資源在 Application layer 已經訂閱了 Control Point 所指定的 Topic，因此往後也不需要再次執行 CROSA 的 PUT(更新)操作，可以直接在 Application layer 對指定 Topic 發佈訊息，就可以對目標資源在 Application layer 上再次進行資源狀態的更動。

### 3.5 CROSA 事件觸發 Eventing

在大多數服務發現協定當中，能對有興趣的服務進行訂閱，當服務發生改變時，可以立即接收到來自訂閱服務端的訊息，在 CROSA 中，我們利用 POST 與 DELET 兩個方法來實作訂閱與取消訂閱兩個功能。在上一節當中，我們提到 CROSA 的控制機制是交由 MQTT 去實際操作，進而減少 Instrumentation Discovery layer 的工作量，另一方面也能透過 MQTT 的 publish/subscribe 訊息傳送模式，提升智慧家庭設備布署上的便利性。然而，因為我們將服務發現協定中，與應用相關的功能提升到 Application layer，也因此在 REST 規範中的 POST 與 DELET 所對應到的訂閱與取消訂閱兩個方法，可以很輕易的在 Application layer 上執行並達成。

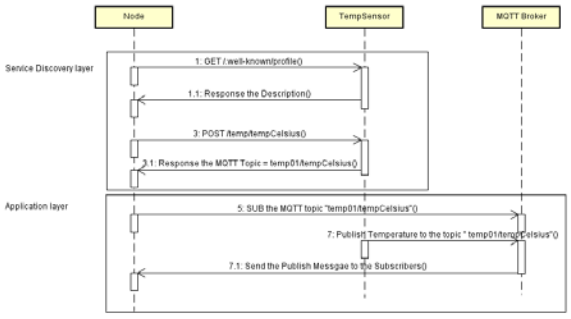


圖 8 透過 CROSA POST 訂閱感興趣資源流程

當某個節點對於某個服務感興趣時，它會透過 CROSA 的服務發現協定取得該服務的 Description，再透過 Description 找到對應的 POST 的方法，去執行訂閱操作，在 CROSA 事件觸發的機制當中，我們也一樣將其實際訂閱的功能提升到 Application layer 進行，CROSA 在這裡所扮演的角色與 CROSA Control 機制一樣，僅提供節點透過 MQTT 訂閱該服務資源的相關資訊。

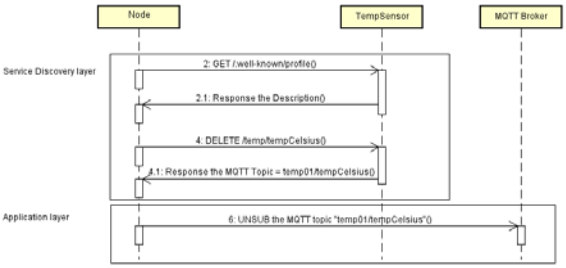


圖 9 透過 CROSA DELETE 取消訂閱資源流程

舉個例子(如圖 8)，若某個節點想持續監聽在受限網路中特定溫度感測器(tempSensor)溫度的變化量時，該節點會透過 CROSA 取得該服務的 Description 後，再針對指定資源透過 CROSA 發送 POST 操作，這時該服務就會回傳該資源在 MQTT 上訊息發佈的 topic 給該節點，此時該節點就會透過 MQTT 訂閱目標資源的 topic。

反之，若該節點不想再持續追蹤該資源的變化時，會透過 CROSA 發送 DELETE 操作來取消對該資源在 Application layer(MUG)上的訂閱。詳細流程如圖 9，在圖 9 當中，假設該節點目前正在訂閱溫度感測器(tempSensor)中 tempCelsius 這個資源；當該節點不想再持會追蹤這個資源時，它會透過 CROSA 的服務發現協定取得該服務的 Description，再透過 Description 找到對應的資源的方法，並執行 DELETE 的操作，這時該服務就會回傳該資源在 Application layer 上訊息發佈的 topic 給該節點此時該節點就會透過 Application layer 取消訂閱目標資源的 topic。

## 四、實驗與數據分析

在這個部分，我們將針對 CROSA 的發現功能做效能評估，實驗對照組將由本實驗室先前研究的 ROSA 服務發現架構[10]做為對照。這個章節裡，我們會評估以下指標數據：(1)特定資源與全部資源的發現時間、(2)控制步驟的封包流量、(3)取得服務描述檔所需流量等，三項目做實驗評估。

### 4.1 實驗設置

我們於一個獨立的無線區域網路做實驗，使用的路由器為 D-LINK Wireless N 300 型號的無線路由器，終端設備為以下：

- 電腦-A(CPU-intel Core i5 2.5GHz、RAM 8G、OS-Win7)
- 電腦-B(CPU-intel Core i5 2.5GHz、RAM 4G、OS-Mac OSX)
- 電腦-C(CPU-intel Core i5 2.5GHz、RAM 4G、OS-Mac OSX)

電腦 A、電腦 B 與電腦 C 分別扮演不同節點，因此在這裡的評估當中，我們的節點數量(n)固定為 3，每個節點最少一個服務提供最少一種資源類型(rt)，且每種資源類型至少有一個資源(m)，因此資源總數最少會有 3 個(m=3)。

接下來我們將設計三個實驗來驗證 CROSA 發

現機制的(1)特定資源與全部資源的發現時間、(2)控制步驟的封包流量、(3)取得服務描述檔所需流量。在第一個實驗當中，我們將要測試 CROSA 發現機制的效率比起 ROSA 的發現機制，是否更為精進。我們將電腦-A、電腦-B 與電腦-C 個別代表不同服務的類型：Light(電燈)、TV(電視)、TEMP(溫度感測器)，並且每個服務類型再分別模擬出 n 個資源，故資源總數 m=n*3(個)。在這個實驗當中，我們分別在三台電腦上再建立一個 Control Point，Control Point 會隨機選擇一種服務類型透過 CROSA 與 ROSA 去發現特定類型的資源，我們將會變動 n 值，去觀察當特定服務類型所擁有的資源數量與每次搜尋的發現時間是否有影響。其中 n 值會從 1 增加至 10。除了觀察特定資源外，我們也會觀察了當 n 值從 1 增加至 10 時，搜尋完所有資源的單一資源搜尋平均時間為何。我們將相同實驗配置，分別在 CROSA 與 ROSA 上測試，藉此比較他們發現的時間。

第二個實驗，我們將測試 ROSA 與 CROSA 在控制資源時所產生控制封包的流量大小。實驗配置部分，電腦-A、電腦-B 與電腦-C 個別代表不同服務的類型：Light(電燈)、TV(電視)、TEMP(溫度感測器)，並且每個服務類型再分別模擬出 10 個資源，故資源總數 30 (個)。在這個實驗當中，我們將分別利用 ROSA 與 CROSA 發現架構，透過 Control Point 發送 GET 與 PUT 請求一次至所有資源，GET 與 PUT 詳細操作內容如下表 2：

表 2 CROSA 實驗二 - 控制操作詳細內容

| 操作類型 / 服務類別 | GET 操作 | PUT 操作 |
|---|---|---|
| Light(電燈) | 電燈狀態 | 控制燈的亮度 |
| TV(電視) | 電視目前頻道狀態 | 轉台 |
| TEMP(溫度感測器) | 目前溫度狀態 | 開/關 |

其中，我們將隨機取 n 個資源分別實作 ROSA 與 CROSA 對應的 GET 與 PUT 操作，重副實驗，變動 n 值由 1 至 10，使得資源總數變多，並觀察 ROSA 與 CROSA 在控制結點時所產生的控制封包流量大小，這個實驗，我們會透過軟體 Wireshark 來監聽封包並計算流量。

第三個實驗，我們將測量 ROSA 與 CROSA 在取得服務描述檔時，所產生的流量大小。實驗配置部分，電腦-A、電腦-B 與電腦-C 個別代表不同服務的類型：Light(電燈)、TV(電視)、TEMP(溫度感測器)，並且每個服務類型再分別模擬出 n 個資源，故資源總數 m=n*3(個)。在這實驗中，我們將會變動 n 值由 1 到 10，並分別利用 ROSA 與 CROSA 發現架構去取得所有資源的描述檔，其間，我們將會透過軟體 Wireshark 來監聽封包並計算流量。

### 4.2 數據分析

**搜尋所有/特定資源所需時間**

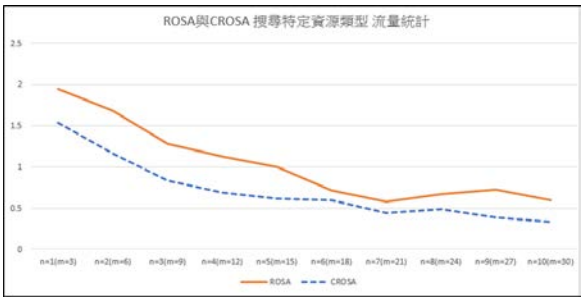


圖 10 ROSA 與 CROSA 搜尋所有資源統計


圖 11 ROSA 與 CROSA 搜尋特定資源統計

在實驗結果(如圖 10 與圖 11)可以看到，在相同配置情況下，CROSA 所表現出來的數據比起 ROSA 所表現出來的數據明顯好得多，其原因在於基於 CoAP 的 CROSA 繼承了 CoAP 在傳輸上的輕量化，比起 ROSA 基於 HTTP 來實作發現，明顯輕巧許多，因此 CROSA 在整體表現下，比起 ROSA 要來的出色。另外，我們可以觀察到，當搜尋特定資源時(圖 10)，資源數越多，所花費時間會變少的原因在於，因為 CROSA 與 ROSA 架構在發送發現請求時，都是透過 Multicast 的方式，也因此為了避免符合請求的資源回應請求端時，在過於相近的時間內發送回應造成封包碰撞，因此 CROSA 與 ROSA 架構都有一個參數 t，來延遲發送時間；本次實驗，我們都將 ROSA 與 CROSA 的 t 值設為 3，其表示，資源回應訊息時，會隨機延遲 3 秒內的時間回應訊息。也因為如此，當單一資源類型有多個數目的資源時，數量越多，則可以更有機會取得更少的隨機延遲時間。另外，圖表中可以看到，在搜尋所有資源時(圖 11)，時間都較為一致，其原因是因為，當要搜尋所有資源時，都必須等到所有資源回應完成，因此消耗時間並沒有因為 n 值增加而減少。

**控制操作封包流量計算**

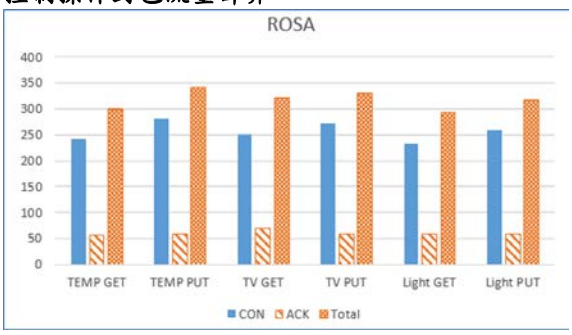


圖 12 ROSA 各類別單一操作封包大小

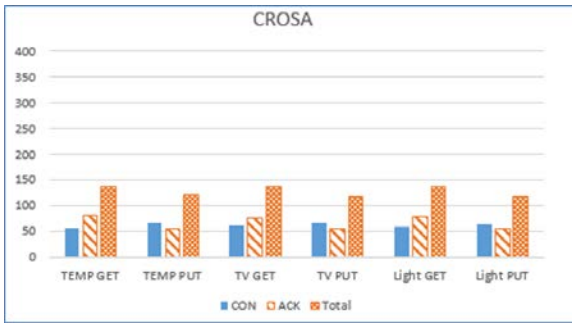


圖 13 CROSA 各類別單一操作封包大小

在執行實驗二以前，我們先分析了各項操作中 ROSA 與 CROSA 的封包大小，如圖 12 與圖 13 所示。我們可以觀察到 CROSA 在控制操作比 ROSA 的控制操作需要較少的流量，其原因在於 CROSA 走的是 CoAP 通訊協定，CoAP 在封包上減少了許多不必要的資訊，也因此，在相同操作封包大小的表現上，使用 CoAP 為基礎的 CROSA，比起已 HTTP 為基礎的 ROSA 來的好很多。

另外，我們看實驗二的結果，圖 14 為 n 由 1 到 10 時 CROSA 與 ROSA 的流量統計；另外，圖 15 為 n 由 1 到 10 時，CROSA 與 ROSA 在流量大小上的差值。由實驗結果可以看出，當操作量越多時，CROSA 能比起 ROSA 節省越多的流量。

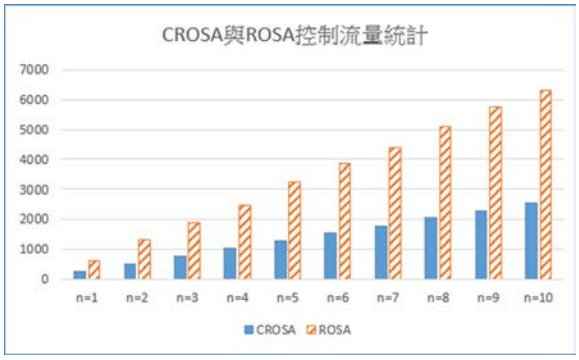


圖 14 CROSA 與 ROSA 控制操作流量統計


圖 15 CROSA 與 ROSA 控制流量差距

**取得服務描述檔流量計算**

在進行實驗三以前，我們先分別針對實驗所執行的三大類型：Light(燈)、TV(電視)、TEMP(溫度感測器)，透過 Wireshark 來測量出 CROSA 與 ROSA 描述檔的流量，如圖 16 所示，我們可以很明顯看到 CROSA 在取得描述檔的流量上比起 ROSA 來得少很多，其實原因在於，在 CROSA 使用的是 JSON 格式作為描述，其中內容也只記載了必要的資訊，且比起 ROSA 省略了許多 XML 格式所需的標籤字元數量(如：<response> 結尾就需要有一個 </response>，在 JSON 中只需要單純的 key-value 的配對)，也因此在描述相同設備服務以及資源的檔案上，CROSA 可以用比較簡單的方式作描述，在傳輸上也就相對輕量許多。

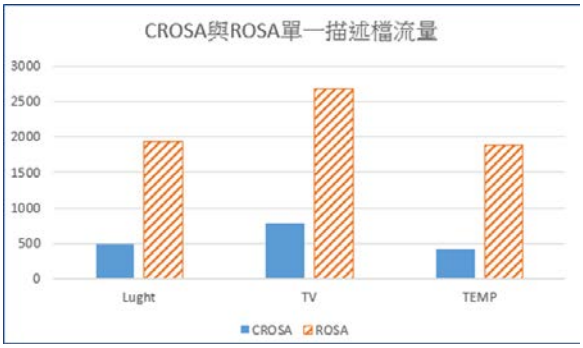


圖 16 CROSA 與 ROSA 取得各服務類型描述流量

再來，我們來看實驗三的結果，圖 17 為實驗三的實驗結果，我們可以很明顯得看到當 n 的數量越大，則 ROSA 與 CROSA 所使用的流量消耗就越大，在圖 18 當中，更可以看出兩者在不同 n 值所差距的流量值。這樣的差距在智慧家庭當中將會非常之大，因為智慧家庭的設備，需要經常向目標設備請求服務與資源的描述檔，當傳輸描述檔的次數越多，CROSA 所能節省的流量就越大，也因此這樣地改善對於有限設備及網路是有非常大的幫助。

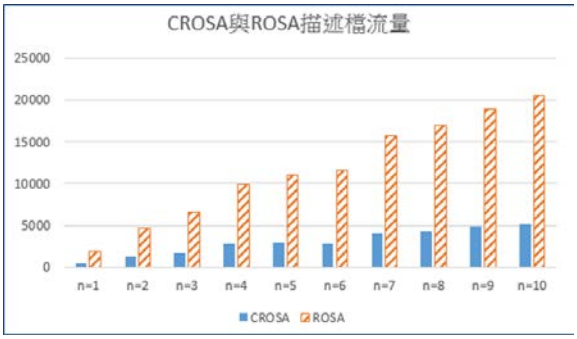


圖 17 CROSA 與 ROSA 隨機取得不同類型描述檔流量

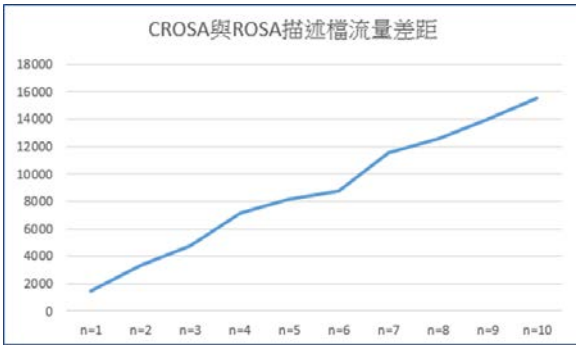


圖 18 CROSA 與 ROSA 隨機取得不同類型描述檔流量差距

最後，本論文以下表 3 來比較 CROSA、ROSA 以及傳統 UPnP 在發現(Discovery)、描述(Description)、控制(Control)與事件(Eventing)各皆段所使用的方法以及 CROSA 改良部分。

表 3 CROSA 與其它方法比較

| | CROSA | ROSA | 傳統 UPnP |
|---|---|---|---|
| 發現 (Discovery) | Multicast/CoAP | Multicast/HTTP | SSDP/UDP |
| 描述 (Description) | JSON | WADL | XML |
| 控制 (Control) | REST | REST | SOAP |
| 事件 (Eventing) | MQTT/TCP | Observer/HTTP | GENA |

## 五、結論

CROSA(Constrained Resource-Oriented Service Administration)以 CoAP 為基礎來實作，並依照發現(Discovery)、描述(Description)、控制(Control)與事件(Eventing)各皆段來實行發現功能，CROSA 改良了 ROSA[10]架構，CROSA 只專注於執行發現(Discovery)與描述(Description)兩個階段的操作，其餘控制(Control)與事件(Eventing)則交由上層應用層 MQTT 來實施，因此降低了在發現階層的負擔。從實驗結果可以發現，在取得 CROSA 的描述檔時，比起取得 ROSA 描述檔時更為節省網路流量；另外，從實驗結果也可以得知，在取得資源的操作方面，基於 CoAP 的 CROSA 架構比起基於 HTTP 協定的 ROSA 架構，也擁有更低的流量開銷。

## 參考文獻


[1] CHEN, Hsiang Wen; LIN, Fuchun Joseph. Converging MQTT resources in ETSI standards based M2M platform. In: Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE. IEEE, 2014. p. 292-295.

[2] COLLINA, Matteo; CORAZZA, Giovanni Emanuele; VANELLI-CORALLI, Alessandro. Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In: Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on. IEEE, 2012. p. 36-41.

[3] GOVINDAN, Kannan; AZAD, Amar Prakash. End-to-end service assurance in IoT MQTT-SN. In: Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE. IEEE, 2015. p. 290-296.

[4] HUNKELER, Urs; TRUONG, Hong Linh; STANFORD-CLARK, Andy. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In:Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on. IEEE, 2008. p. 791-798.

[5] LEE, Shinho, et al. Correlation analysis of MQTT loss and delay according to QoS level. In: Information Networking (ICOIN), 2013 International Conference on. IEEE, 2013. p. 714-717.

[6] LUZURIAGA, Jorge E., et al. A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In: Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE. IEEE, 2015. p. 931-936.

[7] STANFORD-CLARK, A. S.; TRUONG, Hong Linh. MQTT for sensor networks (MQTT-S) protocol specification. International Business Machines Corporation version, 2008, 1.

[8] TANG, Konglong, et al. Design and implementation of push notification system based on the MQTT protocol. In: International Conference on Information Science and Computer Applications (ISCA 2013). 2013.

[9] Transport, MQTT MQ Telemetry. ”V3. 1 Protocol Specification.” (2014).

[10] 陳鵬宇(2014)。智慧家庭中資源導向隨插即用服務管理協定的設計與實現。逢甲大學資訊工程學系碩士班碩士論文。全國博碩士論文資訊網，102FCU05392067。

[11] VILLAVERDE, Berta Carballido, et al. Service discovery protocols for constrained machine-to-machine communications. Communications Surveys & Tutorials, IEEE, 2014, 16.1: 41-60.

[12] SHELBY, Zach; HARTKE, Klaus; BORMANN, Carsten. The constrained application protocol (CoAP). 2014.

[13] ISHAQ, Isam, et al. Group communication in constrained environments using CoAP-based entities. In: Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on. IEEE, 2013. p. 345-350.

[14] CHO, Gunhee, et al. Enhancing CoAP proxy for semantic composition and multicast communication. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers. ACM, 2015. p. 205-208.