結合 gRPC 的 mDNS/DNS-SD 物聯網服務管理機制
Service Management Schemes for IoT Devices based on gRPC and mDNS/DNS-SD

林庭安、廖峻鋒
國立政治大學資訊科學系
Ting-Ann Lin and Chun-Feng Liao
Department of Computer Science, National Chengchi University
{107703005,cfliao}@nccu.edu.tw

摘要

科技日益進步，物聯網 (Internet of Things, IoT) 技術也日漸成熟。越來越多感測器與家用電器等日常用品連上網路形成物聯網。由於大量裝置無人監管，因此需透過服務管理機制整合協調裝置節點共同運作。其中，mDNS/DNS-SD(又稱 Zeroconf) 是一個非常普及、輕量化且彈性高的服務發現協定，然而和其它服務管理機制相比 (如 UPnP)，其完整性不足，尤其是缺乏「服務描述」與「服務存取」機制，目前也較少相關的解決方案被提出。本研究針對此問題，研製能將 gRPC 融合 mDNS/DNS-SD 協定的運作機制，藉此機制補足目前 mDNS/DNS-SD 未臻完備之處，使其能更普及地應用於日常生活之中。

關鍵字：物聯網、服務發現、Zero-configuration、gRPC、mDNS/DNS-SD。

1 前言

隨著資訊科技的蓬勃發展和網路的普及，各式資通訊技術早已和眾人的生活方式密不可分。其中，物聯網 (Internet of Things, IoT) 技術將大量實體物件和網路串聯在一起，這樣具有運算能力的物件稱為智慧物件 (Smart Object)。物聯網的應用廣泛，可與醫療、製造、物流、交通等領域結合。在其眾多應用中，「智慧家庭 (Smart Home)」可說是與一般民眾生活最貼近的一種應用。在「智慧家庭」中，分散各處的軟體和裝置需互相合作以提供能滿足使用者需求的服務 (Service)，包含如何偵測、發現與註冊各式設備、軟體與裝置的功能、如何描述服務元件的功能、如何控制元件、從元件取得資料、推播本身的事件或訂閱其它元件的事件，進而組合多個元件，形成一個服務，這些問題的解決方案通常被稱為「服務管理」機制。。

在各式現有服務管理機制中，UPnP 與 mDNS/DNS-SD 可說是目前最廣被實作與採用的服務管理機制。UPnP(Universal Plug and Play) [1] 目前由 Open Connectivity Foundation 維護，主要被 Intel/Microsoft 推廣，而 mDNS/DNS-SD(Multicast DNS/DNS-based Service Discovery) [2,3] 則已成為 IETF 標準，主要使用於 Apple 產品與部份 Linux 作業系統。無論是 UPnP 或 mDNS/DNS-SD 做為物聯網服務管理機制都有其限制。就 UPnP 來說，過去幾年，我們發現 UPnP 本身應用在智慧家庭環境時在設計上存在一些不足，並嘗試加以改善，包含未支援服務錯誤偵測功能、不完善的服務描述、及效能低落等 [4,5]。UPnP 起初是為家用區域網路所設計，若考量與 LLN 結合，就顯現出更多問題。[6,7] 皆指出，UPnP 使用了大量 XML 格式，而剖析 XML 與儲存 XML 分別需要花費不少計算與儲存資源，例如 [8] 就觀察到 UPnP 節點最高花費超過 3MB 記憶體，在資源受限的嵌入裝置較難實現。此外，UPnP 大量依賴未壓縮且明碼文字的 HTTP 訊息群播，節點數增加到一定程度時效能會驟然低落 [7]。最後，UPnP 本身也被發現具有安全性上的問題 [9]。

相對於 UPnP，目前較多學者認為 mDNS/DNS-SD 有潛力成為 IoT 服務管理機制。mDNS/DNS-SD 重覆利用了原 DNS 的封包格式設計，所以它與現有 DNS 相容，大部份作業系統均有支援，不需額外配置其它系統元件即可使用 [7,10]。另外，它採用 Binary 編碼，對嵌入式系統來說剖析訊息的速度相對於 XML 較快，所需記憶體較低，且 Binary 編碼也有助降低訊息大小，經過優化後亦可在 LLN 使用 [11,12]。然而，和 UPnP 相比，mDNS/DNS-SD 在服務管理功能不如 UPnP 完整：mDNS/DNS-SD 透過五種型態 RRs(Resource Records) 進行服務發現，分別是 PTR、SRV、TXT、A 與 AAAA。PTR RR 封包用於詢問、回覆目前網域中是否有對應的服務型態；SRV RR 封包用於詢問、回覆目標服務的埠號和主機名稱；TXT RR 封包用於詢問、回覆使用者添加的其他資訊；A

RR 與 AAAA RR 封包則用於詢問、回覆主機對應的 IPv6、IPv4 域名。其中並沒有對於服務用途、服務狀態設置描述和控制存取的機制。反觀 UPnP 採用 XML 格式紀錄服務，且將 URI (Uniform Resource Identifier) 提供給客戶端。換句話說，UPnP 服務發現機制不僅可以找到新的裝置與服務，也提供描述檔和存取裝置的方法。由此可見 mDNS/DNS-SD 僅提供部分服務管理功能，並不是一個完整的機制。

服務應用程式介面 (Service API) 是一種處理兩台機器間通訊的技術，可用於解決 mDNS/DNS-SD 缺乏的服務描述與存取機制。服務應用程式介面的種類眾多，常見的包含 REST [13]、SOAP(Simple Object Access Protocol) [14] 和 gRPC(Google Remote Procedure Call) [15] 等。REST 通常基於 HTTP、XML 或 JSON 等通訊協定及標準；SOAP 基於 HTTP 通訊協定並遵從 XML 格式；gRPC 則是使用 HTTP/2.0 通訊協定和二進位的 protocol buffer 訊息格式。與 REST 和 SOAP 相比，gRPC 更為快速且輕量，並支援串流呼叫 (stream calls)。考量到 gRPC 高效能的設計，非常適合用來物聯網環境中，補足 mDNS/DNS-SD 在服務描述與存取機制上的不足。

綜上所述，雖然 mDNS/DNS-SD 相對其他服務管理機制 (如 UPnP) 較有效率且安全，但相較 UPnP，mDNS/DNS-SD 的完整性不足，尤其是欠缺能與之配合的服務描述與服務存取機制。因此，本研究主要的目的即在於以 gRPC 搭配 mDNS/DNS-SD，打造一個既安全、有效率又完整的服務管理機制。期望能藉由此研究所發展的技術，使智慧家庭服務能夠更加普及。

2 相關研究

UPnP 是建構於 HTTP、HTTPU(HTTP Unicast) 和 HTTP-MU(HTTP Multicast over UDP) 之上的服務管理機制，以 SOAP(Simple Object Access Protocol)、GENA(General Event Notification Architecture) 與 SSDP(Simple Service Discovery Protocol) 等應用層協定進行服務呼叫、事件通知與服務發現。儘管 UPnP 相較其他協定完整，且可利用 XML 訊息格式相容於 IMS 等各種服務，但卻面臨許多安全性上的問題。Cui 等人指出 UPnP 協定在不同供應商的閘道器中存在許多安全漏洞，導致中間人攻擊、重送攻擊等威脅 [16]。Nava-Lopez 等人也指出 UPnP 使用的 XML 格式文件具有標準結構，若以明文形式傳送便可能危害機密性、完整性與可用性，並針對此問題提出一個以 Python 編寫的安全工具 [17]。Kayas 等人的研究則指出 UPnP 於事件通知、服務發現、控制等階段皆存在安全漏洞，且提出一個使控制點與伺服器裝置皆可驗證 UPnP 互動的緩解方案 [18]。

mDNS/DNS-SD 由 Multicast DNS 和 DNS Service Discovery 兩種協定搭配而成，是可在單一網域中達到自動設定 (Zero-configuration) 與服務發現 (Discovery Phase) 的物聯網協定。歷來許多學者進行了 mDNS/DNS-SD 協定的相關研究，這些研究大致上可分為二類。第一類為改善 mDNS/DNS-SD 於低功耗網路 (Low-power and Lossy Networks, LLN) 節點應用上的問題。例如 Klauck 和 Kirsche 提出以壓縮封包的方式減少 mDNS/DNS-SD 在 6LoWPAN 中交換的封包數量 [11]。Stolikj 等人則是以代理伺服器機制擴充 mDNS/DNS-SD 協定 [12]。Mahmoud 等人也提出以三種強化機制減少 mDNS/DNS-SD 的封包數量與 CPU 的能量消耗量 [19]。第二類為 mDNS/DNS-SD 結合其他技術的應用。例如 Lin 等人發展了基於 mDNS/DNS-SD 的安全自動配置閘道器結構，用於保護智慧家庭系統免於惡意威脅 [20]。NANTOUME 等人則是將 mDNS/DNS-SD 協定的功能擴展至區域網路以外，提供低成本的電信服務給馬里地區的居民 [21]。然而，目前的相關研究較少著墨於改善 mDNS/DNS-SD 不完整的服務描述與存取機制，而這也是本研究的重點。

3 mDNS/DNS-SD 技術背景

DNS (Domain Name System) [22] 是 mDNS/DNS-SD 的重要基礎，它是一個在分散在全球 IP 網路上，提供網域名稱與 IP 對應查詢服務的分散式資料庫系統。DNS 伺服器分散在全球各地，各伺服器自行維護所管轄的資料，客端連接到任一 DNS 伺服器就可查詢全球的網域名稱與 IP 對應。DNS 伺服器與客端通訊方式和 HTTP 十分類似，屬於無狀態的 Query-Response 模式，若客端欲實現有狀態的連續互動 (Session)，須自行儲存訊息的 Transaction ID(紀錄於 DNS Header 中) 並加以對應。DNS 訊息格式如圖 1所示，值得注意的是，DNS 無論 Query 或 Response 訊息格式皆採用一致的格式，除了表頭外，分為四大區:Questions、Answer、Authority 與 Additional。客端進行 Query 和 DNS 伺服器進行 Response 時，所用到的區段 (Sections) 有些許不同。如圖 1中的 Query，使用到 Questions 與 Additional 區段，而 Response 時則會用到 Answer、Authority 與 Additional 區段。Questions 區段主要和 Query 配合，包含不定長度的 query name 欄位與固定 4 bytes 的 Info 欄位。Info 欄位中可指定 query type，在本論文討論範圍中會用到的 query type 包含 A/AAAA 用來進行域名-IP 對應、PTR 用來查詢特定服務類型的節點和 SRV 用來查詢節點的 IP 與 Port 資訊。一旦 Query 送出，DNS Server 就會進行 Response。由圖 1中可知 AnswerAnswer、Authority 與 Additional 區段和 Questions 區段

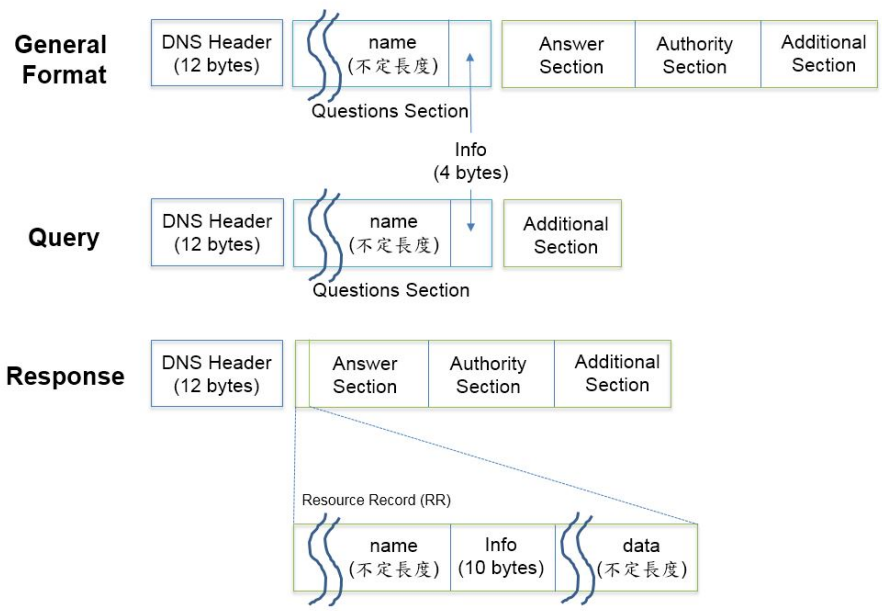

圖 1: DNS 訊息格式 (RFC 1035): 最上方為通用格式，中間為 Query 用到的區段，下方為 Response 使用區段。Additional Section 存放 Query 或 Response 時所需附帶的額外資訊。

最大的不同在於這三個區段多了不定長度的 data 欄位，用來裝載回應資訊。一組匹配的 Query/Response 訊息有相同的 name 和 Transaction ID，因此 client 即使收到很多 Responses，也可根據此資訊判斷那些 Responses 是用來回應那些 Query。

在 IoT 網路 (如家庭網路) 中，不太可能由使用者自行架設 DNS 伺服器，mDNS [2] 可以在沒有 DNS 伺服器的情況下，提供區域性的名稱-IP 對應服務 (主要是管理 <name>.local 的網域名稱)。mDNS 採用 IP Multicast 機制，所有在區網內的節點可透過對特定群播位址 (224.0.0.251:5353 或 [FF02::FB]:5353) 進行 Query/Response 來取得名稱對應服務，區網內所有節點都同時扮演 DNS Client 與 Server 角色，因此都會收到 Query，也回應 Answer。一般情況下只有符合 query name 中指定的節點才需要回應。群播容易造成網路癱瘓，因此 mDNS 規格書中有不少部份在說明如何透過快取與撤消 (Known-Answer Suppression) 機制來降低它對網路效能的影響，此外，規格書中對於 DNS 訊息的長度也建議不要超過 255 bytes。

DNS-SD [3] 基於 mDNS 的群播規則與 DNS 的訊息格式實現了服務發現 (Service Discovery) 的功能。其實現的服務發現功能為: 在沒有中央註冊伺服器的情況下，透過標準的 DNS Query/Response 訊息格式與 mDNS 的群播機制，找到具備某些服務類型 (Service Type) 節點的 Local 網域名稱及其 IP and Port，這裡值得一提的是，DNS-SD 所指的服務類型是 Internet Services (如 FTP、SMTP 等應用層協定)。然而，在 IoT 環境中，服務發現機制所指的服務類型並非 Internet Services，而是指提供特定功能的服務/裝置，例如 Light 指的是任何可發出光源的裝置，在 IoT 環境下，都可由 HTTP 所存取，則可定義這類裝置的服務類型為 **_Light._sub._http._udp.local**，其中，附加在 _http 之前的 _Light._sub 代表 BinaryLight 這種 Sub Type。因此，在 DNS-SD 中的 Service Type 其意義為應用層通訊協定，而 DNS-SD 的 Service Sub Type 指的才是一般 IoT 系統中的 Service Type。

4 系統設計

本研究主旨為設計一個兼具效率與完整性的服務管理機制。本節將先說明 mDNS/DNS-SD 提供的服務現機制，分析其問題，接著說明本研究設計以 gRPC 結合 mDNS/DNS-SD 的實作方法。

4.1 設計理念分析

mDNS/DNS-SD 以 IP 網路位置 224.0.0.251 作為群播位址，並透過 5353 埠將封包群播至區域網路中。在區域網路中的 mDNS 節點具有 Query 和 Response 兩種功能，Query 用於尋找有興趣的節點，Response 則是由符合 Query 的節點在回覆時使用。DNS-SD 則定義了如何命名和排列 DNS 紀錄，如服務描述和封包區段用途等。其中對於服務的描述大致上可分為 Instance、Service Type 和 Domain 三個部分，以 <Instance>.<ServiceType>.<Domain> 格式表示。而封包區段則有 PTR、SRV、TXT、AAAA 與 A 五種主要紀錄型態。

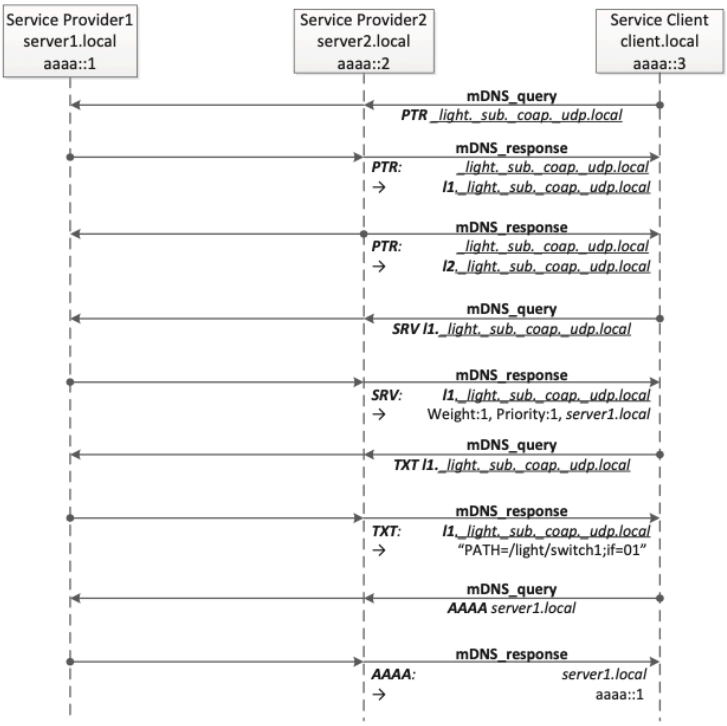

圖 2: mDNS/DNS-SD 服務發現時序圖

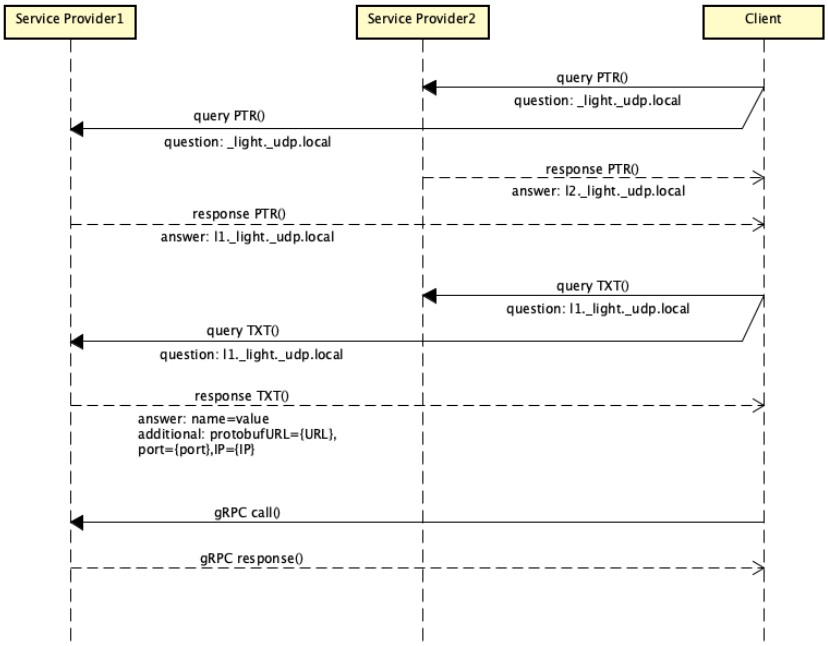

圖 3: gRPC 結合 mDNS/DNS-SD 服務發現時序圖

以尋找 light 服務為例，當客戶端欲尋找有興趣的 light 服務時，首先須使用 PTR RR 封包作詢問，若此時有節點的服務型態與之對應，便會將 Instance 以群播的方式回覆給區域網路。取得 Instance 後，便可使用 SRV 和 TXT RR 封包詢問服務的埠號和主機名稱等資訊。最後可透過 SRV RR 封包回覆的 Domain Name，並使用 A 與 AAAA RR 封包詢問主機對應的 IPv6 和 IPv4 域名 (如圖 2)。我們發現，mDNS/DNS-SD 提供的服務描述僅限於技術層面（如服務類型、服務所使用的協定等），因此僅能提供服務發現功能。意即客戶端僅能透過 mDNS/DNS-SD 找到感興趣的服務及其相關資訊，卻無法實際使用該服務。以上述尋找 light 服務為例，客戶端僅能透過 mDNS/DNS-SD 得知網域內存在的 light 服務與其埠號、主機名稱等資訊，但無法得知此服務的操作方法。

現今已有許多成熟的服務應用程式介面 (Service API) 可用於解決上述問題。REST、SOAP 和 gRPC 皆是常見的服務應用程式介面，以下進一步對三者做簡單的分析與比較：

1. REST(REpresentational State Transfer) 是一種軟體建構風格，符合此架構風格的服務能夠在網路中互相傳遞資訊。REST 通常基於 HTTP、URI、XML 或 JSON 等常見的通訊協定及標準，並使用 GET、POST、PUT、DELETE 方法操作資源。

2. SOAP(Simple Object Access Protocol) 是使用在網頁服務中交換資料的一種協定，基於 HTTP 通訊協定並遵從 XML 格式。UPnP 採用此協定作為服務存取機制。

3. gRPC(Google Remote Procedure Call) 是由 Google 開發且開源的 RPC(Remote Procedure Call) 框架，使用 HTTP/2.0 通訊協定和二進位的 protocol buffer 訊息格式，並支援跨語言的一元呼叫 (unary calls) 和串流呼叫 (stream calls)。

由上可知，gRPC 使用 protocol buffer 作為訊息格式，二進位的序列化訊息格式相較於 REST 和 SOAP 使用的 JSON、XML 格式更簡單、快速且輕量。此外，由於 gRPC 使用 HTTP/2.0 作為通訊協定，因此額外支援雙向串流呼叫 (Bidirectional streaming call) 方法。gRPC 共具備四種服務方法：

1. 一元呼叫 (Unary RPCs)：如同一般的函式呼叫，客戶端向伺服器發送單個請求，並獲得單個回應。

2. 伺服器串流呼叫 (Server streaming RPCs)：客戶端向伺服器發送單個請求，並獲得單個串流。客戶端會從得到的串流中讀取訊息，直到沒有更多訊息為止。

3. 客戶端串流呼叫 (Client streaming RPCs)：客戶端將訊息寫入串流中，並在寫入結束後將串流發送給伺服器。伺服器會從得到的串流中讀取訊息並回覆單一回應給客戶端。

4. 雙向串流呼叫 (Bidirectional streaming RPCs)：客戶端和伺服器雙方共同使用兩個讀寫串流發送訊息。由於兩個串流是獨立運行的，因此客戶端和伺服器可以任意順序進行讀寫。

其中，gRPC 的串流呼叫方法可使服務控制更有效率。以監測溫濕度感應器為例，當客戶端希望不斷獲取伺服器端感測器的量測值時，若使用一元呼叫，客戶端需在每次欲獲取量測值時發送請求並建立與伺服器間的連結。相對地，若使用伺服器端串流呼叫，客戶端只需向伺服器端發送單個請求，便可建立一長久的連結以獲得一連串量測值，降低通訊所花費的時間。gRPC 具有快速、輕量、有效率等優點，因此本研究藉由結合 gRPC 與 mDNS/DNS-SD，解決 mDNS/DNS-SD 完整性不足的問題，設計一個更加完善的服務管理機制。

4.2 模組設計

針對上述問題，此步驟將要透過引進 gRPC 以設計出能讓客戶端完成服務發現且能夠存取服務的服務管理機制。根據 mDNS 規格書，mDNS 封包有 Question、Answer 和 Additional 三個區塊。Question 區塊用於向其他節點發出請求、Answer 區塊用於註冊網路或回應其他節點、Additional 區塊則是用於補充 Answer 區塊的資訊。在 DNS 使用的五種主要紀錄中，TXT RR 用於傳送使用者添加的其他資訊，因此本研究將透過 TXT RR 的 Additional 區塊，以 Key=Value 的方式將可取得 protocol buffer 的 URL 與服務的埠號、主機對應的域名等資訊夾帶在其中提供給客戶端。本研究的系統運行如圖 3所示。客戶端以群播的方式將 PTR RR 送至區域網路內的每個節點，符合服務型態的節點會將 Instance 回傳。透過群播 PTR RR，客戶端可取得網域內所有節點的 Instance，並利用 Instance 選定欲存取的對象。接著再利用 TXT RR 取得可獲取 protocol buffer 的 URL 與服務的埠號、主機對應域名。透過 URL 取得 protocol buffer 後，便可使用 protocol buffer 產生其所使用語言的標準程式碼，並透過生成的標準程式碼，以一元呼叫、客戶端串流呼叫、伺服器端串流呼叫、雙向串流呼叫等方法取用服務。

由於此設計是由 mDNS/DNS-SD 既有的發現機制衍伸，在進行 TXT Response 時將 URL 與其他資訊添加至 Additional 區塊。對於不支援此設計的客戶端和裝置仍為傳統的 mDNS/DNS-SD 服務發現機制，故與傳統的 mDNS/DNS-SD 互為相容 (如圖 4)。

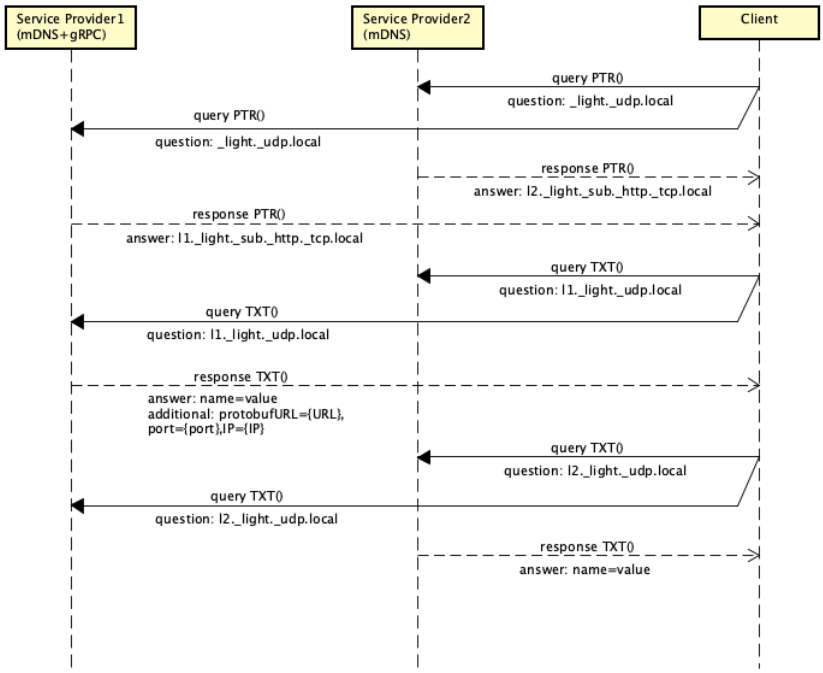

圖 4: 搭配 gRPC 與未搭配 gRPC 之 mDNS 節點服務發現時序圖

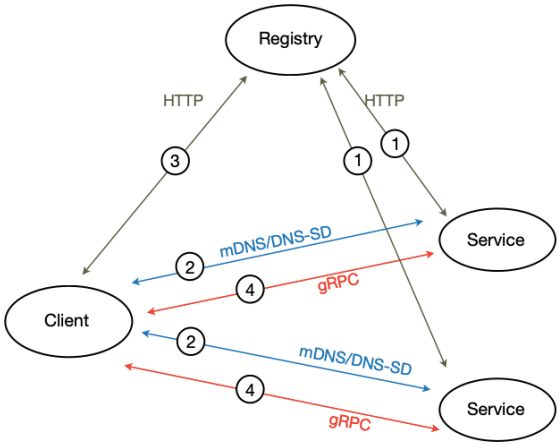

圖 5: 服務發現及存取流程架構圖

5 系統實作與效能評估

前文提及，本研究主要貢獻為基於 gRPC 服務應用程式介面，搭配 mDNS/DNS-SD 設計出能使客戶端完整地發現並取用服務的服務管理機制。本節將詳述系統實作內容並透過實驗評估此系統。

5.1 系統實作

本研究基於 Node.js 開發平台，以 Github 社群提供的開放原始碼專案 multicast-dns 為基礎實作。客戶端透過 multicast-dns 專案向網域內所有服務發送請求並接收回應以完成服務發現功能。為達成服務描述與存取機制，還需實作一集中 Registry 以存放各項服務提供的服務描述 proto 檔。此 Registry 以非關聯式資料庫 MongoDB 實作，並透過 Github 社群提供的開放原始碼專案 rest-on-mongo 在 MongoDB 上建構 RESTful 應用程式介面。客戶端透過 multicast-dns 向服務取得 URL 後，可簡單地經由 HTTP 向 Registry 拿取服務的 proto 檔。取得服務描述檔後，再以 Google 提供的 node.js gRPC library 存取服務。

圖 5為完整的服務發現與服務請求流程。首先，服務在開始運行前，需先透過 RESTful 介面將服務描述檔儲存至 Registry 中 (圖 5，步驟 1)。接下來，當客戶端欲存取服務時，需先透過群播 PTR RR 取得服務的 Instance，再透過得到的 Instance 發送 TXT RR 並得到 URL 與埠號等資訊。(圖 5，步驟 2)。之後，透過 URL 向 Registry 取得感興趣的服務描述檔。(圖

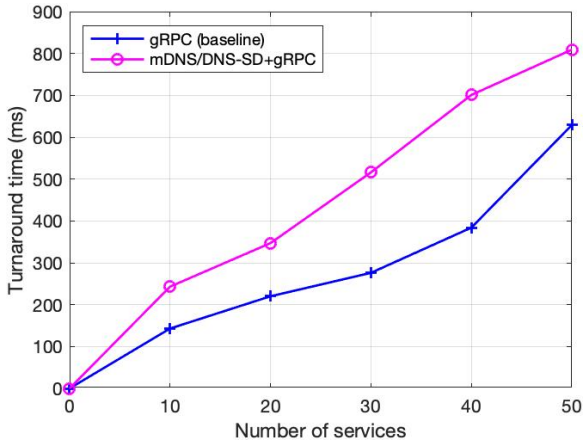

圖 6: Turnaround time 實驗

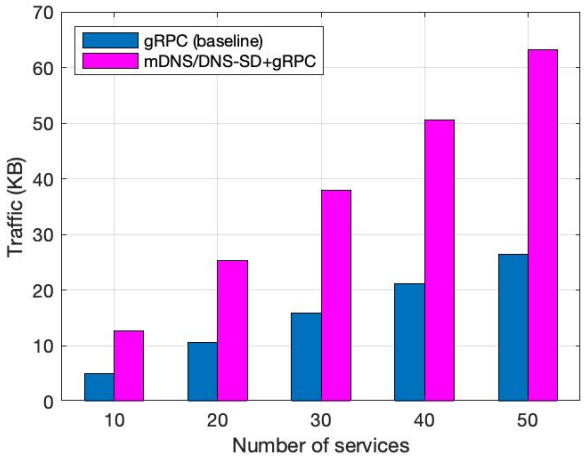

圖 7: 交通量實驗

5，步驟 3)。最後，便可透過 gRPC 存取服務。(圖 5，步驟 4)。

5.2 效能評估

在典型的智慧家庭中，網域內通常擁有許多服務，請求、響應的訊息傳遞也相當頻繁。本節實驗模擬智慧家庭中的開關燈服務，比較僅使用 gRPC 與使用 mDNS/DNS-SD 搭配 gRPC 的情形，針對時間和交通量進行測試。

在 turnaround time 實驗中，我們採用 Javascript 中的 Date 物件進行測量。在實驗中，我們分別設置 10、20、30、40、50 個開關燈服務，並將匹配率固定設為 30%，測量從客戶端發送請求至實際取用服務並得到回傳值的時間。測試結果如圖 6所示，從圖 6可知，十字的線條為僅使用 gRPC 的情形，可視為 Baseline，而使用 mDNS/DNS-SD 搭配 gRPC 所花費的時間約為其兩倍。不過，在僅使用 gRPC 的情況下，客戶端必須事先知道網域中所有服務的埠號、服務描述檔等資訊，才可進行服務存取。儘管以 mDNS/DNS-SD 進行服務發現將造成額外負擔，卻可以此實現零組態服務管理機制。且在服務數量為 50 的範圍內所造成的 overhead 皆不超過 0.5 秒，因此我們認為此效能負擔尚在可接受的範圍之內。

在交通量測試中，我們採用 Wireshark 封包量測套裝軟體進行量測。在此實驗中同樣設置 10、20、30、40、50 個開關燈服務，且將匹配率固定設為 30%，測量從客戶端發送請求至實際取用服務並得到回傳值所需之交通量 (packet size * number of packets)。測試結果如圖 7所示，圖 7中可觀察出服務數量與交通量成線性成長，而使用 mDNS/DNS-SD 搭配 gRPC 所需的交通量約為僅使用 gRPC 的二點五倍。

由於 mDNS/DNS-SD 服務發現機制需向網域中的所有服務以群播方式發送請求，接收到請求的服務也以群播方式向網域中所有節點發送回應，儘管本研究僅使用 mDNS/DNS-SD 五種 RR 中的 PTR RR 與 TXT RR，仍產生不少交通量。而使用集中 Registry 存取服務描述檔也造成系統較高的負擔。

6 結論

歷來許多學者進行了 mDNS/DNS-SD 協定的相關研究，但較少著墨於改善 mDNS/DNS-SD 不完整的服務描述與存取機制。在本篇論文中，我們提出一個基於 gRPC 服務應用程式介面的擴充機制，使 mDNS/DNS-SD 功能更加完備。檢視初步設計後，我們發現目前的設計在效能上仍有提升的可能性，因此未來將嘗試透過 Klauck 和 Kirsche 提出的封包壓縮方法減少 mDNS/DNS-SD 的交通量 [11]，並嘗試以點對點方法使客戶端可以更輕量的方式取得服務描述檔，期望能使 mDNS/DNS-SD 更普及地應用於日常生活中。

參考文獻

[1] A. Presser, L. Farrell, D. Kemp, and W. Lupton, “Upnp device architecture 1.1,” in *UPnP Forum*, vol. 22, 2008.

[2] S. Cheshire and M. Krochmal, “Multicast dns,” Tech. Rep., 2013.

[3] ——, “Dns-based service discovery,” Tech. Rep., 2013.

[4] C.-F. Liao, Y.-W. Jong, and L.-C. Fu, “Toward reliable service management in message-oriented pervasive systems,” *IEEE Transactions on Services Computing*, vol. 4, no. 3, pp. 183–195, 2011.

[5] C.-F. Liao, H.-C. Chang, and L.-C. Fu, “Message-efficient service management schemes for mom-based upnp networks,” *IEEE Transactions on*

Services Computing, vol. 6, no. 2, pp. 214–226, 2013.

[6] I. Al-Mejibli and M. Colley, "Evaluating transmission time of service discovery protocols by using ns2 simulator," in *Wireless Advanced (WiAD), 2010 6th Conference on*. IEEE, 2010, pp. 1–6.

[7] B. C. Villaverde, R. de Paz Alberola, A. J. Jara, S. Fedor, S. K. Das, and D. Pesch, "Service discovery protocols for constrained machine-to-machine communications." *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 41–60, 2014.

[8] H. Hayakawa, T. Koita, and K. Sato, "Evaluation of sonica compared with upnp and jini," *Proceedings of ICMU 2006*, 2006.

[9] D.-s. Lim, J.-y. Kang, and I.-w. Joe, "A sdn-based network intrusion detection system to overcome upnp security drawbacks," in *Mobile and Wireless Technologies 2016*. Springer, 2016, pp. 117–126.

[10] M. Mahyoub, A. Mahmoud, and T. Sheltami, "An optimized discovery mechanism for smart objects in iot," in *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2017 8th IEEE Annual*. IEEE, 2017, pp. 649–655.

[11] R. Klauck and M. Kirsche, "Enhanced dns message compression-optimizing mdns/dns-sd for the use in 6lowpans," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 596–601.

[12] M. Stolikj, R. Verhoeven, P. J. Cuijpers, and J. J. Lukkien, "Proxy support for service discovery using mdns/dns-sd in low power networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*. IEEE, 2014, pp. 1–6.

[13] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[14] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple object access protocol (soap) 1.1," 2000.

[15] K. Indrasiri and D. Kuruppu, *gRPC: Up and Running: Building Cloud Native Applications with Go and Java for Docker and Kubernetes*. O'Reilly Media, 2020.

[16] B. Cui, Q. Zhang, X. Zhang, and T. Guo, "Research on upnp protocol security of gateway device," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2017, pp. 450–458.

[17] I. Nava-Lopez, L. Prudente-Tixteco, J. Olivares-Mercado, G. Sanchez-Perez, K. Toscano-Medina, and L. C. Castro-Madrid, "Security tool for upnp protocol on smart tv," in *2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE, 2019, pp. 1–6.

[18] G. Kayas, M. Hossain, J. Payton, and S. R. Islam, "An overview of upnp-based iot security: Threats, vulnerabilities, and prospective solutions," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2020, pp. 0452–0460.

[19] A. Mahmoud, M. Mahyoub, T. Sheltami, and M. Abu-Amara, "Traffic-aware auto-configuration protocol for service oriented low-power and lossy networks in iot," *Wireless Networks*, vol. 25, no. 7, pp. 4231–4246, 2019.

[20] H. Lin, D. S. Kim, and N. W. Bergmann, "Saga: Secure auto-configurable gateway architecture for smart home," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2019, pp. 11–1109.

[21] A. NANTOUME, B. M. DEGBOE, B. NIANG, A. D. KORA, and S. OUYA, "Design and deployment of a low-cost communication solution in rural areas: case of the central region in mali."

[22] P. Mockapetris, "Rfc 1035—domain names—implementation and specification, november 1987," *URL http://www. ietf. org/rfc/rfc1035. txt*, 1987.