國立臺北大學電機工程學系專題報告

車載乙太網路診斷

Diagnostic over Internet Protocol (DoIP)

組員:林建澄
指導老師:陳永源 老師
執行期間:110 年 09 月至 111 年 06 月

1. 摘要

本計畫實作符合 ISO13400-2:2019 之 DoIP 車載乙太網路診斷系統。透過開源程式碼，我們將符合標準之服務端和自行開發的 ECU 程式載入樹莓派，並透過 PC 的自製客戶端去和 Server 通訊，使得訊息可以在 Client、Server、Electronic Control Unit(ECU)三者之間通訊，並順利完成診斷服務。而整體的通訊流程皆必須符合 ISO13400-2 的規範。除此之外，我們也將客戶端程式製作成圖形化介面，使得使用者可以更有效率的使用系統、分析診斷結果。

關鍵字:DoIP、Socket、車載網路

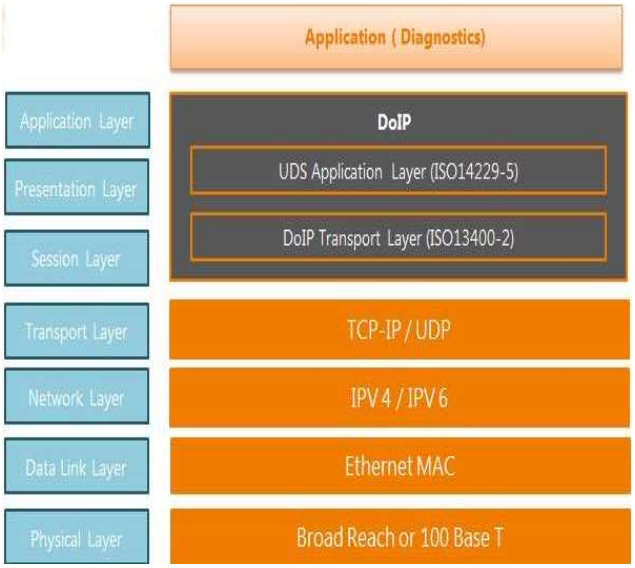
2. 簡介

現今車輛內的電子電機設備日趨複雜，隨之而來的，車輛內部彼此之間的通訊量也會上升，傳統上，車輛內通訊主要是依靠 CAN Bus、FlexRay 等方法來通訊，而當前越來越多車廠引進車載乙太網路的方法來解決此問題。透過乙太網路，可以使整體的通訊負載量大幅上升，而我們本次的專題利用乙太網路的方法去實現符合標準的規範的車輛診斷功能，將車輛當作 server 端，其底下可能包含多個電子控制單元(ECU)節點，我們可以透過 client 端去對 server 請求功能執行或訊息回覆，當 client 端接收到 server 回傳的資訊後，就可對其進行分析，並讓診斷技師做出相對應的操作。

3. 專題進行方式

甲、 DoIP 標準文件與網路架構

當應用程式需要透過網路傳輸資料時，必須建立一套規範讓雙方的硬體和軟體設備遵守，才能順利完成通訊。常見的網路架構有 OSI 網路七層和網路協定，而協定的出現讓軟硬體在需要進行通訊時，有能夠遵守的準則。為了能在車載乙太網路上實現車輛診斷功能，DoIP 是建立於 TCP/IP 網路協定的基礎上而定義出的規範，其主要包括傳輸層的 ISO13400-2 以及應用層的 ISO14229-5。ISO13400-2 定義多項基於 TCP 和 UDP 的服務，而 ISO14229-5 主要用於制定各式的診斷服務功能，而此篇報告將於 3-(丁)中著重討論 ISO13400-2:2019。



圖一、DoIP 網路架構

乙、 DoIP 系統架構

DoIP 的系統架構可分為車輛內部網路和車輛外部網路，如圖二所示。

內部網路主要有以下節點:

A. Edge gateway:即為 Server 端，是車輛唯一對外接口，車輛的診斷訊息皆必須由此節點進出，其會將外部診斷設備的資訊進行解析後，將封包傳至指定的位置。

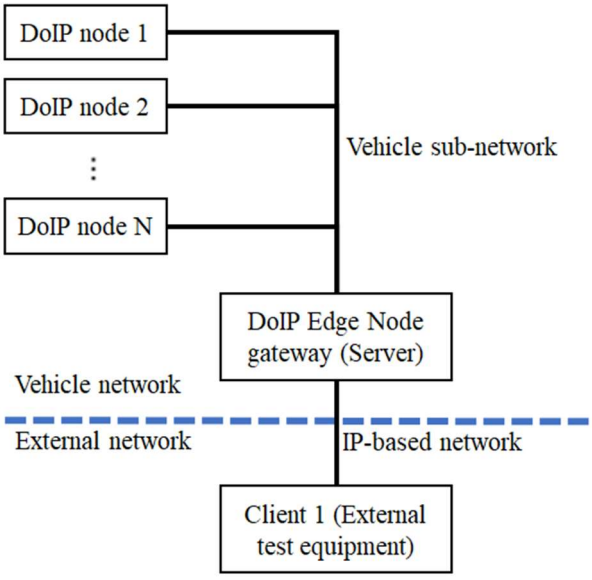
B. Gateway:當車輛內部的某些節點的通訊是不符合 DoIP 的規範時，就必須利用 gateway 進行封包的轉換，將資訊轉成符合內部節點的通訊格式，再進行診斷。

C. ECU:為車輛內部的電子控制單元，如剎車系統、電池偵測系統等，其所使用的通訊方法可能是不符合 DoIP 的標準規範，如 CAN Bus、Flexray。

D. DoIP node:內部節點所使用的通訊方式已符合 DoIP 的規範。

外部網路主要有:

A. 外部診斷設備: 即為 Client 端，主要用於向車輛請求診斷服務，執行診斷功能，其會和車輛端的 edge gateway 進行連線，並向欲診斷的 ECU 或 DoIP node 發送診斷訊息。



圖二、DoIP 系統架構

丙、 DoIP 通訊流程

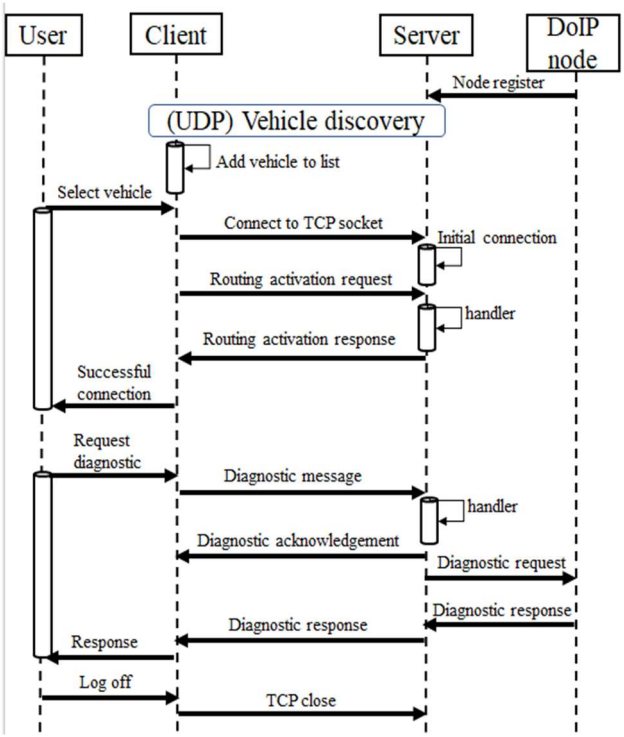
圖三說明車輛進行診斷時所經過的流程，圖由左到右分別代表使用者、Client 外部診斷設備、Server Edge gateway、車輛內部節點。

首先，Client 和 server 間必須透過 UDP 的方式進行 vehicle discovery，這個車輛發現的動作讓 client 端將 server 的車輛基本資訊儲存起來，以便操作診斷設備的人使用。

第二步是讓操作的人選擇車輛進行連線，在 DoIP 的規範下，除了進行基礎的 TCP/IP 連線外，還必須執行 TCP 的 Routing activation request 路由請求，順利完成此任務後，client 和 server 才算連線成功。

第三步是使用者向車輛端發送診斷訊息，執行診斷功能，client 會向 server 表明其來源地址、所欲診斷的目標以及診斷資訊，而 server 端收到後會將訊息轉傳至相對應的位置執行 diagnostic request，並在收到內部節點的回應後，將訊息回傳給外部診斷設備。

最後，當使用者完成診斷任務時，client 會將連線的 TCP socket 關閉，即中斷連線。



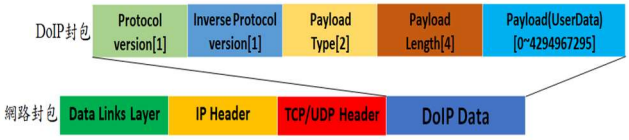
圖三、DoIP 通訊流程

丁、 DoIP 主要功能及其格式介紹

DoIP 於傳輸層和應用層中的規範分別為 ISO13400-2、ISO14229-5，此處將主要討論前者所提供的功能，而將簡略介紹後者所提供的診斷服務。

在進入標準文件的功能介紹之前，我們先解說 DoIP 的基本格式規範，如圖四。其訊息位於 TCP/UDP Header 後方，主要由五項元素所組成: (中括號內代表所含 Byte 數)

A. Protocol version: 代表版本資訊，由開發商所制定。
B. Inverse protocol version: 和 Protocol version 互補，兩者的 Byte 相加為 0xFF。
C. Payload type: 表示本筆資料的類型，用 2 Bytes 表示。
D. Payload length: 代表本筆資料的長度，用 4 Bytes 表示，資料長度最長可為 4294967295Bytes。
E. Payload (user data): 資料內容，包含 source address(SA)、target address(TA)以及診斷訊息等，不同的內容都有相對應的長度規範，如 SA 和 TA 為 2Bytes。



圖四、網路封包和 DoIP 基本封包格式

ISO13400-2 定義於傳輸層之中，其功能主要基於 TCP 和 UDP 兩種傳輸模式，此規範中所使用到的功能主要有以下幾項:

A. (UDP) Vehicle identification request:

此功能用於車輛發現，client 端可以向網路發送 UDP 廣播，請求位於網路內的 server 進行回應，當 server 收到後，回應內容包含 Vehicle Identification Number (VIN)、Logical address、Entity ID(EID)、Group ID(GID)，這些資訊皆有助於 client 端設備辨識車輛代碼，其 payload type、payload length、payload 所使用參數可參考文件所述，並於此報告章節 4-(丙)展示實驗結果。

B. (TCP) Routing activation request:

Routing activation 用於建立 TCP/IP 基礎連線之後，client 端必須在發送一筆請求，向 server 表明 source address(SA)、activation type，若其 SA 符合規定，則連線順利完成，反之則失敗。

C. (TCP) Alive check:

每個 server 都有其允許的最大連線數，當 server 的連線已滿，且又有新的 client 連線請求時，server 就會向所有已連線的 client 發送 alive check request，若診斷設備沒有在規定時間內回覆訊息，則連線將會被 server 端關閉，反之，則能繼續保持連線狀態。

D. (TCP) Diagnostic message

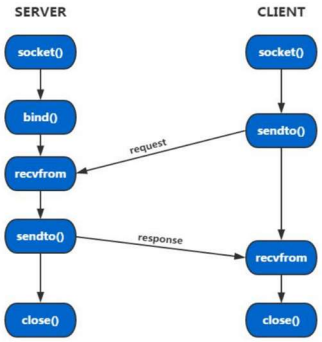
建立完連線後，當外部設備想要診斷某 ECU 或 DoIP node 節點時，就會使用此功能，訊息格式包含 SA、TA 以及診斷資訊，TA 可能是車輛內部任一節點的 logical address，而診斷訊息的內容必須是 server 端有提供的服務，定義於 ISO14229 中，若 TA 或診斷訊息輸入錯誤，則會導致 server 端無法順利解析，因而導致則診斷失敗。

ISO14229-1、ISO14229-5 提供很多應用層的診斷服務，我們將在 4-(丙)實做符合 ISO13400-2 之診斷系統，並加入三項應用層服務，分別為 Diagnostic session control、Read data by identifier、 Write data by identifier，在這三項功能終，我們可以瞭解應用層服務如何在 client 和 server 之間做傳輸、解析、執行。

戊、 Socket 網路通訊介紹

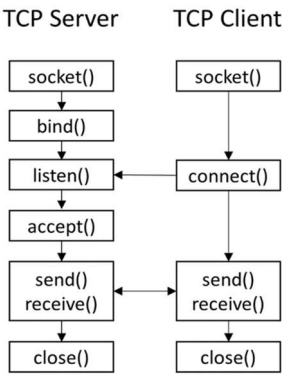
在進入成果展示之前，我們先簡單回顧電腦如何透過網路向另一部機器傳遞訊息，主要的實作方式即為 socket 網路通訊，有了 socket，我們不必從零開始建立，只要透過 socket API 直接調用網路服務即可，而網路服務分為 TCP 和 UDP，我們所需要關注的兩個主要參數為 IP address、Port，以下為兩者通訊的流程介紹。

圖五是 socket udp 的通訊流程，其傳輸訊息時，server 和 client 並沒有建立連線，server 只有利用 bind()監聽網路中某些 IP address 或 Port 是否有訊息丟出，因此若是處於不穩定的網路環境，有可能會有掉封包的問題，導致訊息無法順利傳給對方。



圖五、Socket UDP

圖六是 socket tcp 的實作流程圖，兩者通訊前，server 先處於 listen()的監聽環境，而 client 端必須進行 connect()的動作，在 server 端 accept()後，才是順利完成連線，並且能開始互相傳遞資訊。



圖六、Socket TCP

4. 主要成果

甲、 簡述

本次專題我們的實作方式為使用符合 ISO13400-2 之 server 端開源程式碼，一開始先對其做測試驗證，確認此程式是否符合標準文件所規範。接著，我們便自行開發符合規範的 Client 端外部診斷設備，並將其製作成 UI 介面，讓使用者能更輕易的操作，而我們也將 client 程式設計的更智慧化，使得未來使用者若需要自行新增符合 ISO14229 的應用層診斷功能，不必修改程式碼，只需要在設定功能的.xml 檔中按照標準格式輸入，即可完成新增功能的動作。最後，我們也製作一個 ECU 程式，其擁有三項 ISO14229 的應用層診斷服務功能，如此一來，整個診斷功能就有了一定的完整度。以下我將針對 client 端的程式設計架構以及整體系統整合後的執行結果進行介紹。

乙、 DoIP 之 client 端實作

A. 開發環境

開發工具: PC

開發環境: Linux Ubuntu 20.04

程式語言: C 語言

特別函式(庫)介紹:

GTK library:

用來提供 C 語言圖形化介面開發的開源函式庫。

Sys/socket.h:

Linux 下的網路通訊 API 函式庫，提供 TCP 和 UDP 兩種服務可使用。

Select():

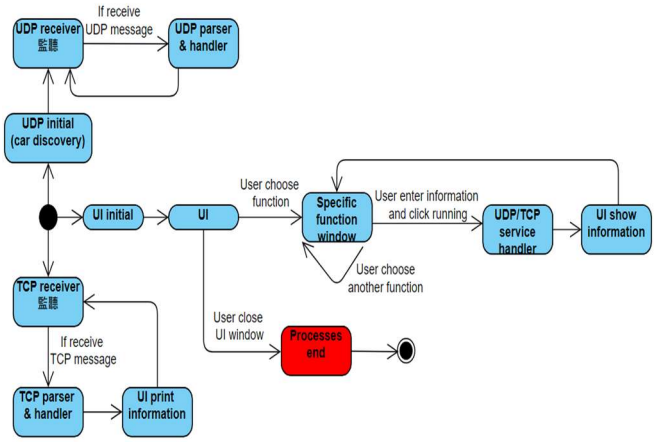
將原本 blocking 的 socket 通訊服務轉換成 nonblocking 的模式。

Pthread():

此函式用於 multithread 功能開發，使得程式程序能平行化執行。

B. 程式架構

將外部診斷設備製作成 multithread 的程式，可以更有效率的執行任務，因此，我們將 client 端程式分成三個 thread，分別為 udp receiver、tcp receiver 以及 UI 視窗介面，前兩者主要屬於後端處理，而 UI 視窗則為前端介面，圖七為整個程式的有限狀態機。



圖七、Client 端 finite state machine

C. 利用.xml 檔智慧化功能

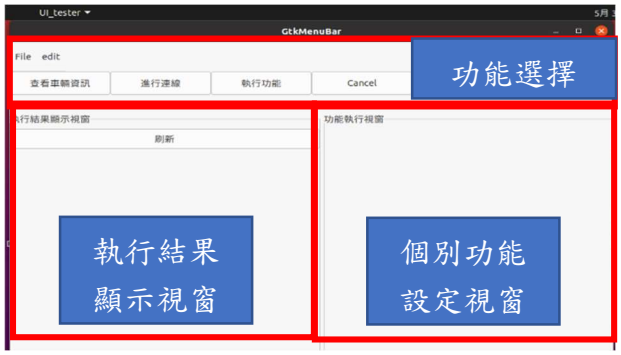
透過讀取 xml 檔的方式，我們將 target address 以及 ISO14229 所提供的診斷服務製作成 xml 格式，使用者可以直接在此檔案中新增、刪除功能，並在 client 端執行後，直接利用 UI 視窗中的選單選取所需的服務，就不必手動輸入每筆資料。

```
<External_tester_uds>
    <Target_address>
        <ta>
            <ta_address>0130</ta_address>
            <ta_name>addres is 0130</ta_name>
        </ta>
    </Target_address>

    <Service>
        <Service_ID>10</Service_ID>
        <Service_name>(0x10)Diagnostic session control</Service_name>
        <Subfunction>
            <name>Defult session</name>
            <id>01<id>
            <param_len>0</param_len>
        </Subfunction>
        <Subfunction>
            <name>Programming session</name>
            <id>02<id>
            <param_len>0</param_len>
        </Subfunction>
        <Subfunction>
            <name>Extended diagnostic session</name>
            <id>03<id>
            <param_len>0</param_len>
        </Subfunction>
    </Service>
```

圖八、xml 檔設定格式

D. Client 端展示



圖九、client 端視窗

丙、 系統整合及執行結果

本次實驗所使用的系統架構如圖十，我們使用一台 PC 作為外部診斷設備，並透過兩台樹莓派模擬車輛(server 端)以及其內部節點，並透過 wifi 機，client 端可向 server 發送、接收訊息，並完成整個診斷服務，以下幾點將展示每個功能的執行結果。

圖十、實驗系統架構

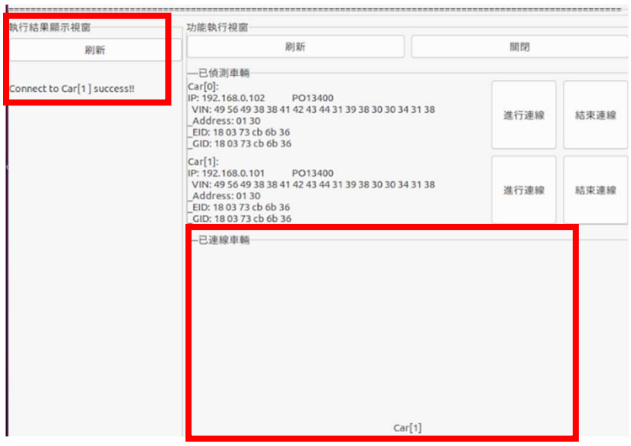
A. Vehicle discovery

Client 端程式執行後，將自動發送一次 vehicle discovery 請求，而使用者點擊”查看車輛資訊”功能後，可以查看每個 server 的資訊，並且可以自行選擇車輛進行連線。

圖十一、Vehicle discovery 執行結果

B. Routing activation

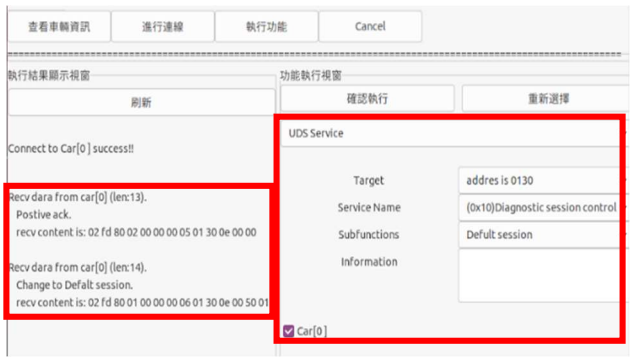
當使用者於 UI 視窗中點擊”進行連線”時，client 會向 server 發送 TCP/IP 連線請求，並執行 routing activation 的功能，若是成功連線，則 UI 視窗會告知使用者並顯示出來，並在右下方會顯示目前所有已連線的車輛，圖十二為向 car[1]進行連線後之執行結果。



圖十二、Routing activation 執行結果

C. Diagnostic message

使用者點選”執行功能”後，可以選擇要執行的 ISO14229 診斷服務以及需要被診斷的車輛內部節點，此處所出現的服務皆定義在 xml 設定檔之中。在選擇完診斷服務、子功能並點擊”確認執行”後，使用者會收到兩筆訊息，分別為診斷狀態以及診斷結果，圖十三以 ISO14229 中的 Diagnostic session control 服務為例，使用者收到第一筆回傳為 positive ack，而第二筆為 change to default session，代表所選的服務成功執行。



圖十三、診斷功能執行結果

D. Alive check

若設定 server 的最大連線數為 1，若有一 client 1 已向此 server 連線，而此時又有另一 client 2 希望連線，則 server 會做 alive check 的動作，client 1 必須向 server 做出回覆才能繼續保持連線，而 client 2 則會連線失敗，如圖十四所示。反之，若 client 1 沒有回覆，則會被中斷連線且 client 2 連線成功。

圖十四、Client 1(左)、Client 2(右)連線結果

5. 結論

本次專題一開始我們一步一步慢慢了解 DoIP 相關資訊，並研究相關標準，包括 ISO13400-2、ISO14229 等，接著我們找到相關能使用的資源，即符合 ISO13400-2 之 server 端開源程式碼，在完成測試驗證的程序之後，我們也自行開發出擁有三項 ISO14229 服務之 DoIP node，並透過自製的圖形化的外部診斷程式，讓使用者能更完整體驗整個 DoIP 診斷服務，最後透過系統整合，使得各程式之間能順利通訊，完整功能的執行、展示。

6. 參考文獻

[1] ISO 13400, “Diagnostic communication between test equipment and vehicle over Internet Protocol (DoIP)”, 2019.

[2] ISO14229, “Unified diagnostic services (UDS)” , 2006.