國立陽明交通大學
資訊科學與工程研究所
碩士論文

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University
Master Thesis

一個低延遲與高流量 5G 核心網路之設計與實作

Design and Implementation of a Low-Latency and High-Throughput 5G Core Network

研究生：李家安（Lee, Chia-An）
指導教授：陳志成（Chen, Jyh-Cheng）

中華民國一百一十年六月
June, 2021

一個低延遲與高流量 5G 核心網路之設計與實作
Design and Implementation of a Low-Latency and High-Throughput 5G Core Network

研 究 生：李家安　　Student: Chia-An Lee

指導教授：陳志成 博士　　Advisor: Dr. Jyh-Cheng Chen

國立陽明交通大學
資訊科學與工程研究所
碩士論文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Yang Ming Chiao Tung University
in Partial Fulfilment of the Requirements
for the Degree of
Master of Science

June 2021

Taiwan, Republic of China

中 華 民 國 一 百 一 十 年 六 月

國 立 陽 明 交 通 大 學

博碩士論文紙本暨電子檔著作權授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之學位論文，為本人於國立陽明交通大學資訊科學與工程研究所 丙組， 109 學年度第二學期取得碩士學位之論文。

論文題目：一個低延遲與高流量 5G 核心網路之設計與實作
指導教授：陳志成

一、紙本論文授權

紙本論文依著作權法第15條第2項第3款之規定辦理，「依學位授予法撰寫之碩士、博士論文，著作人已取得學位者…推定著作人同意公開發表其著作」。

二、論文電子檔授權

本人授權將本著作以非專屬、無償授權國立陽明交通大學、台灣聯合大學系統圖書館及國家圖書館。

論文全文上載網路公開之範圍及時間：	
中英文摘要	必需公開
本校及台灣聯合大學系統區域網路	■ 中華民國 115 年 7 月 1 日公開
校外網際網路及國家圖書館	■ 中華民國 115 年 7 月 1 日公開

說明: 基於推動「資源共享、互惠合作」之理念與回饋社會及學術研究之目的，得不限地域、時間與次數，以紙本、光碟或數位化等各種方式收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

授權人：_____（親筆簽名）

中華民國 110 年 07 月 02 日

國立陽明交通大學碩士學位論文審定同意書

資訊科學與工程研究所　李家安 君

所提之論文

題目：一個低延遲與高流量 5G 核心網路之設計與實作

Design and Implementation of a Low-Latency and High-Throughput 5G Core Network

經學位考試委員會審查通過，特此證明。

學位考試委員會（簽名）

口試委員：

連俐（召集人）　陳志成

張宏鈞

論文已完成修改

指導教授　陳志成　（簽名）

所　　長　易志偉　（簽名）

中華民國 110 年 06 月 24 日

誌　謝

首先我要感謝我的指導教授陳志成教授，從我還是懵懂無知的專題生時，便已經給予我許多教導與建議。在就讀碩士班的兩年期間，老師不但提供了扎實的碩士訓練，讓我們從尋找問題、觀察現象、提出解決方法、驗證實作學習了做研究的方法與態度，也會在我們遇到問到問題時不厭其煩的與我們討論並給予建議。同時，老師也提供了我們充足的實驗資源與設備，並透過與產業密切的合作，讓我們提早學習到產品開發過程中團隊開發、溝通合作、計劃訂定與實行的技巧。

其次我要感謝 free5GC 的團隊夥伴，包含陳斯傑博士、邱德治工程師、翁甫廉工程師，以及參與開啓專案的學長姐們，包含奕華、冠穎、訓頡、家佐、張霽、啓恩、達魯學長，以及瑋庭學姊，在軟體開發初期，所有人一起討論問題、互相協助，才有 free5GC 現在的架構，在學長姐們的帶領下，我們才能比較快速的瞭解核心網路並進入開發狀態。尤其是特別感謝帶領團隊的陳斯傑學長，給予團隊正確的方向，在團隊毫無頭緒時提供我們方向，並且每每在我們對標準不甚理解時，都能引導我們正確的瞭解標準內容。同時，我也想感謝張宏鉦研究員以及謝承穎學長，在我撰寫論文時給予許多建議與幫助，讓我在遇到問題時可以快速的找到解決方法並予之突破。

再來我想感謝的是實驗室的夥伴彥傑、耀文、浚于、亮瑜、惟鈞，不管是課業上遇到問題時，或是計劃上碰到瓶頸，這群夥伴們總是不離不棄的給予我協助、一同討論、一起面對問題。除了物理上的協助外，當我遇到挫折時，也願意聽我訴說苦水，給予我精神上的鼓勵。另外我也想感謝學弟妹們，尤其是浩澤學弟願意協助我在論文研究上的實驗，以及胤年學弟願意接手實驗室網管這項艱鉅的任務。

最後我想感謝我的家人，能讓我毫無後顧之憂的學習、鑽研知識、進行研究，是家人給予我支持與鼓勵，才能讓我度過研究時遇到的種種壓力，讓我順利完成這篇論文。

一個低延遲與高流量 5G 核心網路之設計與實作

學生：李家安　　指導教授：陳志成 教授

國立陽明交通大學 資訊科學與工程研究所

摘　　要

隨著行動網路演進至今，有越來越多功能需要滿足，第五代行動網路 (5G Cellular Networks) 也因應而生。5G 行動網路的三大訴求有 1. 增強行動寬頻通訊 (eMBB) 2. 超高可靠度低延遲通訊 (URLLC) 3. 大規模物聯網通訊 (mMTC)，而藉由網路功能虛擬化的特性 (Network Function Virtualization, NFV)，5G 核心網路可以更加方便操作、快速部屬、無痛擴展。然而現如今的核心網路架構是否眞的能達到上訴三大訴求？還是會爲網路功能虛擬化所帶來的代價而犧牲了高流量低延遲？

本篇論文針對高流量低延遲問題，從觀察使用情境出發，提出一套節點內降低控制端 (control plane) 通訊延遲、增加用户端 (user plane) 通訊流量的核心網路架構並予之實作。在不失網路功能虛擬化特性下，透過 DPDK 與 Shared memory 的操作，小心的達到訊息溝通零記憶體複製 (zero-copy) 通訊，同時對於用户端提供規則快速查找、高流量跨節點轉發功能。

最後針對此核心網路進行效能分析，驗證可以達到超過傳統 5G 核心網路專案 11 倍的用户端流量與低於傳統一半的控制端延遲。

關鍵字：蜂巢式網路、5G、核心網路、網路功能虛擬化、高頻寬、低延遲

Design and Implementation of a Low-Latency and High-Throughput 5G Core Network

Student: Chia-An Lee Advisor: Prof. Jyh-Cheng Chen

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University

Abstract

As the cellular evolved, there is more and more feature must be matched in core network. That's why it comes with 5G cellular networks. In 5G, there are 3 main requirements: 1. Enhanced Mobile Broadband (eMBB) 2. Ultra-Reliable and Low Latency Communications (URLLC) 3. Massive Machine Type Communications (MMTC) Besides, network function virtualization (NFV) are also getting more and more popular in this age. With the features of NFV, 5G core network can be easy to operate, fast to deploy, and painless to scaled. However, is the architecture of 5G core network already meet the 3 requirements nowadays? Will it suffer from high throughput and low latency only with the feature of NFV?

This thesis with focus on the problems of high throughput and low latency. We start from some scenario observation, propose an inter-communication solution to low down the control plane communication delay and improve the user plane traffic forwarding throughput. We also implement the proof of concept to this solution and provide a new core network implementation. Without losing the benefit of NFV, using DPDK and shared memory, we carefully rich zero-copy control plane communication. We also provide fast rule lookup with high throughput forwarding in user plane.

In the end, we evaluation our 5G core and proof our user traffic can rich 11 times more efficient than origin. And our control plane latency can have one over two times then the origin one as well.

Keywords: Cellular Networks, 5G, Core Network, NFV, High Throughput, Low Latency

目　錄

圖 目 錄

表 目 錄

一、 緒論

1.1 5G 概觀

行動網路的發展至今，以然數十年有餘，概約每十年變會進行一次跳躍性的革新。從最初的 1G 與 2G，著重於語音的通訊，並企圖達到移動通訊的能力，讓人們可以隨時隨地的互相通訊。進入了 3G 的時代後，行動網路加入了封包交換 (packet switching) 的能力，使得網際網路透過手機與行動網路進入大眾的手上。隨著智慧型手機的普及，透過手機與網路接觸的使用體驗大幅增加，人類對行動網路頻寬的需求日益不足，4G 的出現便是爲了彌補這樣的需求，從此人們得以享有穩定的行動網路服務。

縱觀 1G 到 4G 的演進，行動網路再再爲了滿足人類的使用而演進。然而到了 5G 的時代，行動網路卻已不再僅僅侷限於個人用户的需求，而是企圖符合更多的使用場景，達到「萬物聯網」的野心。爲滿足如此龐大的野心，過往的架構將不再適用於此，因此有了 5G 的出現。

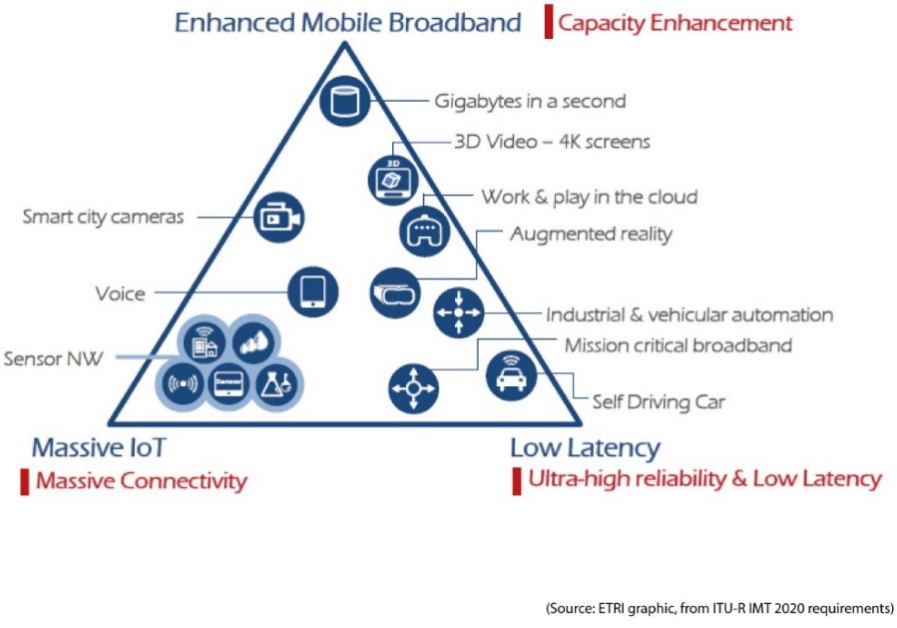


圖 1.1: 行動網路 2020 及未來之應用場景 [1]

5G 全名爲第五代行動通訊網路，系指繼 4G 之後由 3GPP 組織所定義出的最新行動網路通訊技術，爲滿足更多種類的情景與應用，國際行動通訊組織 (International Mobile

Telecommunications, IMT) 於國際電信聯盟無線通訊部門 (International Telecommunication Union Radiocommunication Sector, ITU-R) 制定 IMT-2020 [1]，希望以技術的角度，提出 5G 網路的三大願景：1. 增強行動寬頻通訊 (enhanced Mobile Broadband, eMBB)：指希望增強行動網路的頻寬，達到理論值 $20Gbps$ 下行與 $10Gbps$ 的上行，用以符合逐漸增加的 AR/VR 應用、4K/8K 影像串流等。2. 超高可靠度低延遲通訊 (Ultra-Reliable and Low Latency Communications, URLLC)：提供低於 10^{-5} 的網路傳輸錯誤率，與低於 $1ms$ 網路延遲服務品質，支援如遠端醫療、車聯網、高精密工業加工等應用。3. 大規模物聯網通訊 (massive Machine Type Communications, mMTC)：預期需服務超過 10^6 個裝置同時連線上網的需求，使未來 IoT、工業 4.0 的場景可支援超高密集連線通訊。

爲因應如此多元且靈活的網路特性，過往 4G 的專用網路架構以然不再適合，如何設計出得以適應如此多場景應用的網路架構，便是未來 5G 所要面臨的挑戰。

1.2 研究動機與貢獻

隨著 5G 的發展，爲符合電信商與專有網路需求，行動網路也逐漸向 cloud-native 的趨勢靠攏。透過網路功能虛擬化 (virtualization)、軟體化 (softwarization)，網路功能可以以軟體甚至是容器的方式進行部屬，大大提高行動網路部屬上的擴展性 (scalability)。但是有別於 4G 專業設備的部屬方式，以軟體形式部屬勢必無法達到硬體加速的效能，如此一來如何能達到增強頻寬的效果？另外目前 (R15) 的 5G 網路架構並未能滿足低延遲的需求，尤其在 NF 與 SBA 的設計之下，控制層反而需要通過比以往 4G 更多的分工處理才能完成，且雖然在網路功能結構上有所改變，但在流程上還是可以看到大量 4G 流程的影子，這樣的設計能否因應未來應用上劇烈的挑戰？最後，隨著更多元的設備連接，控制層訊號的數量將遠超以往，該如何解決控制層通訊延遲與壅塞的狀況？本篇論文針對上訴的問題，提出了以下幾點改變與嘗試：

- 我們利用具有彈性及擴展性的網路功能虛擬化 (Network Function Virtualization, NFV) 平臺，設計了於同一節點上更有效率的核心網路溝通方式。
- 透過 DPDK 與共享記憶體的框架，重新設計與實作 SBI 與 N4 介面。透過共享記憶體，提供零記憶體複製 (zero-copy) 的節點內控制訊號通訊方式以減少訊號延遲，並藉由 DPDK 提供跨節點高流量的用户端轉發。

- 我們重新審視 3GPP 的流程，發現在換手流程上有不必要行爲造成用户端路徑增加，同時我們也提出了修改方法。

總結而言，本篇論文提出並實作了提高流量並降低延遲的 5G 核心網路架構。

二、 背景與挑戰

行動網路主要分成接入端網路 (AN, access network) 與核心網路 (CN, core network)，其中接入端最常見的形式便是以基地臺 (base station) 爲主的無線網路接入 (RAN, radio access network)，主要負責聯接使用者設備 (UE, user equipment) 與核心網路、訊號解碼、波束成型、協助核網進行移動管理等等功能，包含 3G 時期的 nodeB、4G 時的 eNodeB、與 5G 時的 gNodeB。核心網路則是負責處理除了接入端外剩下所有事務，包含認證、行動管理、服務品質管理、封包轉送、用量計費、連線與路由規劃、使用者資料紀錄等等功能，包含 2/3G 時的 GPRS 核心網路、4G 的 EPC、與 5G 的 5GC。而核心網路之外會接入資料網路 (DN, data network)，包含但不限於我們常見的網際網路 (the Internet) 或電信商可能會提供的 MEC 網路。完整架構可參考圖 2.1，此爲 3GPP 於 23.501 [2] 標準定義之 5G 行動網路架構圖。

本章我們會著墨於 5G 核心網路的介紹，包含用户層與控制層的介紹，以及常見流程，然後介紹並比較幾個知名的 5G 核心網路專案，最後會描述目前 5G 核心網路所遇到的挑戰。

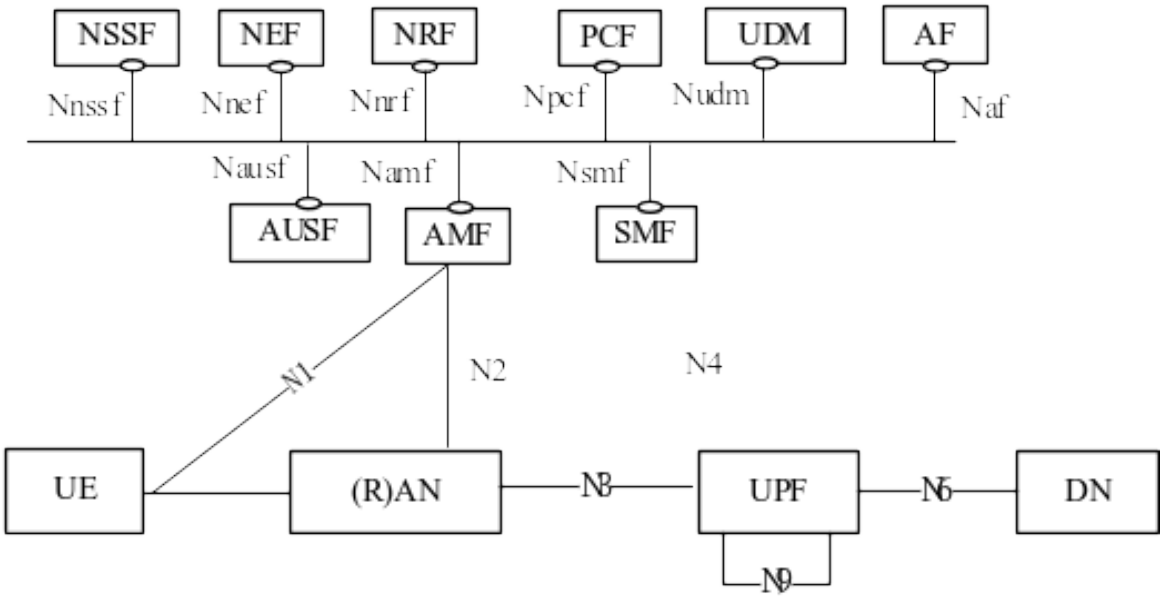


圖 2.1: 5G 核網架構 (取自 TS 23.501 Figure 4.2.3-1 [2])

2.1 5G 核心網路架構

要瞭解 5G 核心網路首先要對 4G 核心網路有一些概念，在 4G 時代，核心網路主要有幾個重要的部件 (component)，Mobility Management Entity (MME)、Serving Gateway (S-GW)、PDN Gateway (P-GW)、Home Subscriber Server (HSS)、與 Policy and Charging Rules Function (PCRF) (見圖 2.2，這裡不討論其他額外功能)，MME 負責連線與移動的管理，S-GW 負責與 RAN 對接，將 LTE 的承載 (bearer) 轉換成 Internet 的形式，P-GW 負責與網際網路間接，HSS 會記載使用者資料與幫助認證，PCRF 則是作規則與計價的管理。而隨著 4G 行動網路的發展，3GPP 會議與電信商越加越多新的功能後，發現此架構不再適用於新功能的發展，決定對核心網路有架構性的改變，進而發展出了 5G 核心網路。5G 核心網路在架構上有幾個突破性的改變，包含：

- **網路功能虛擬化 (NFV, Network Function Virtualization):** 核心網路爲因應雲端 (cloud) 與微服務 (microservice) 的架構，逐漸往網路功能虛擬化方向發展，其所有功能部件皆以軟體形式呈現以方便於在標準化硬體 (commodity hardware) 上部屬，並且核心網路亦將 4G 的五大部件依據功能性，拆分爲更小的單位，並以 Network Function 稱之 (以下簡稱 NF)。
- **用户端與控制端分離 (CUPS, Control and User Plane Separation):** 4G 時如 S/P-GW 同時具備用户端功能與控制端功能，但在逐漸發展後，3GPP 認爲應該將用户端與控制端分離，於是研議出 4.5G 時的 CUPS，將 S/P-GW 的用户端功能拉出並以 PFCP (Packet Forwarding Control Protocol) 協定控制，而 5G 時更是把 S/P-GW 的使用者功能合併並獨立拉出成一個 NF。
- **服務導向架構 (SBA, Service Base Architecture):** 有鑑於 5G 將核網區分爲更多小單位的 NF，如此多種不同種類的 NF 若需互相溝通，將需要開發出多種不同的介面與協定而導致開發與擴展困難，因此 3GPP 於 5G 時採用 SBA 架構，其介面統稱爲 SBI (Service Base Interface)，透過制定統一的通訊格式，減少需要對每個介面開發出新協定的繁瑣，並且若需要新增介面或 NF，僅需重新使用相同格式便可輕鬆開發出新的介面或 NF。
- **網路切片 (Network Slice):** 由於 5G 希望提供更多的網路功能給使用者使用，如何限制功能與功能間的接觸、將相似或上下依賴的功能分群、以及將網路功能擴充或縮減以

有效利用資源便是以此而生的問題。而網路切片便是提供幫助網路功能分群的服務，透過網路切片的概念，可以讓管理者用切片爲單位，更方便的管理大量的網路服務。

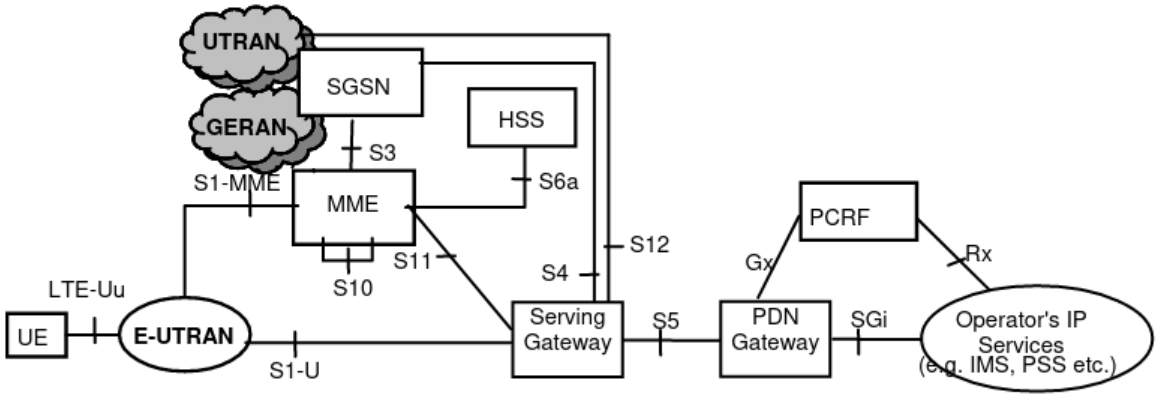


圖 2.2: 4G 核網架構 (取自 TS 23.401 Figure 4.2.1-1 [3])

爲因應以上幾個架構性改變，核心網路被重新定義成多個 NF。3GPP 會議於 Release-15 時至少定義了 23 種 NF (詳見 TS 23.501 6.2 章)，我們取其中較爲重要的 9 個 NF 來介紹：

NF	**描述**
AMF	Access Management Function，主要負責註冊、接入、以及移動管理等，會透過 N2 介面與 RAN 溝通，此介面使用 NGAP 協定，並且透過 N1 介面間接與 UE 溝通，以 NAS 協定承載。
AUSF	Authentication Server Function，負責 3GPP 接入與 non-3GPP 接入的認證流程。
NRF	Network Repository Function，負責幫助服務探訪 (service discovery)、有效的 NF 資訊維護與提供、NF 健康狀態維護、NF 註冊、更新、取消註冊的狀態通知等功能。
NSSF	Network Slice Selection Function，選擇服務 UE 的網路切片 (network slice) 集合，維護、設定、決定 NSSAI 與 S-NSSAI 的關係映射。
PCF	Policy Control Function，負責提供控制端所需的管理規則，並提供統一的規則管理框架以保護網路功能行爲。

SMF	Session Management Function，處理包含連線管理 (包含 AN 與 UPF)、UE IP 的發放管理、流量導向轉換與分流 (traffic steering)、虛擬網路 (VN, virtual network) 群組管理、協助執行管理與收費規則、協助品質管理 (QoS) 與緩衝管理 (buffering)、漫遊等等功能。
UDM	Unified Data Management，負責 5G AKA 憑證產生、使用者識別、隱私保護的識別碼 (SUCI) 解碼、UE 的服務 NF 註冊管理、訂閱資訊管理等功能。
UDR	Unified Data Repository，負責儲存資料，包含 UDM 的訂閱資料、PCF 的規則資料、應用服務資料、NF 群組資料、需要被接觸的結構化資料等。
UPF	User Plane Function，是核網中唯一的用户端 NF，基本上負責執行 SMF 所下的指令，可能包含封包路由與轉送、封包封裝與解封、與外部連接的 PDU 連線、流量規則執行與使用報告、服務品質管理 (QoS)、上下行緩衝、終止符號 (End Marker) 的傳送等等功能。

表 2.1: NF 介紹

在這些 NF 中，又以 AMF、SMF、與 UPF 最常被提出討論，主要是因爲 AMF 是負責與 RAN 的控制端訊號對接的，幾乎所有 UE 與 RAN 所發出的控制訊息都要透過 AMF 接入核心網路，且大部分訊息也都是 AMF 會進行處理或初步解譯；UPF 是所有用户端封包一定會經過的，若 UPF 的效率不彰將導致整個網路使用起來遲緩、無法達到 5G 訴求的高流量低延遲；而 SMF 則是扮演決定用户端路徑流量、與分析控制端訊號並轉換成用户端規則的重要角色，可謂是控制端與用户端規則互相轉換的重要元件。

在架構上，5G 亦有分 SA (Standalone) 與 NSA (Non-Standalone) 兩種架構，所謂 SA 是指純 5G 架構，包含 RAN 使用 5G 之 gNodeB 且 CN 使用 5G CN，而 NSA 則是爲了因應跨世代的向下相容，可支援 5G CN 配合 4G RAN 或 4G CN 配合 5G RAN。就目前大部分商用 5G 行動網路都是採 NSA 架構，但會希望漸漸往 SA 架構轉移。

2.1.1 5G 核心網路控制層介紹

不同於 4G 將所有功能至於五個部件上，5G 核心網路將所有功能拆分成不同 NF，並且可以明確的定義哪些 NF 是屬於控制端的 NF 而那些是屬於用户端，基本上除了

UPF 外的 NF 都是屬於控制端。相較於 4G，由於 5G 需要承載更大量的設備、支援高速移動、功能拆分溝通、更多複雜的應用等，控制端的訊息在設計起來時便要考慮如何降低其延遲性、更容易被解讀、且一體化，因此有了如上所訴的 SBA 架構設計。3GPP 於 SBI 上採用了 HTTP/2 的協定通訊，相對於 HTTP/1，HTTP/2 透過資料壓縮與多路複用等能力，大幅減少網路的延遲。另外於 HTTP/2 協定內採用 RESTful API 的設計，透過有效的 HTTP Verb 使用，提供簡潔易辨識的 URI 資源操作，而 3GPP 更是採用 OpenAPI 格式，可以輕鬆快速的定義人類易讀的 YAML 格式，並透過轉換工具 [4] 產生程式碼。

除了 SBA 的設計外，核心網路在 N2 及 N3 介面保留使用了 4G 的協定以向下相容，其中 N2 介面是介於 AMF 與 RAN 之間，在此介面上採用 NGAP 協定，是 4G 時 MME 與 RAN 溝通所使用的 S1AP 協定的新版，其內容規格大致相同採用 TLV 設計，另外在 N2 之上 UE 與 AMF 乃至於 SMF 的溝通格式採用 NAS (non-access stratum) 訊息，其中 non-access 是相對於 RAN 與 AMF 直接相接的 AS (access) 所命名的。而 N4 介面則是介於 SMF 與 UPF 之間，使用 4G 於 R13 CUPS 時採用的 PFCP (Packet Forwarding Control Protocol) 作爲通訊協定，同樣承襲了 TLV 設計，並透過五種規則資訊元素 (IE, Information Element) 對用户端進行規則下達。

2.1.2 5G 核心網路用户層介紹

在用户端 5G 比 4G 簡化許多，僅使用 UPF 作爲用户端之唯一 NF。UPF 不會自行決定規則，亦不會與基地臺或其他 UPF 溝通來做判斷，而是完全遵守 SMF 透過 N4 介面下達的規則，因此隨著連線數量漸多、穩定及延遲需求漸高，此介面的溝通效率便越發重要。N4 介面目前使用 PFCP (Packet Forwarding Control Protocol) 協定，此協定遵守 TLV 格式，亦即以 TLV 爲一個單位，每個單位內部會包含三個資訊：1. 種類 (T, type) 2. 長度 (L, length) 3. 值 (V, value) ，並且在值內可以另外包含一個或多個更小單位，而透過使用 TLV 格式可有效壓縮訊息長度，增加網路頻寬使用效率。

而在 PFCP 訊息裡，以 PFCP 連線 (PFCP Session) 爲單位，基本上會與 PDU 連線互相映射。在 PFCP 連線內會包入封包處理規則，這些規則主要分爲五類：

- **PDR (Packet Detection Rules):** 主要負責辨別那些進入封包屬於這個規則內。
- **FAR (Forwarding Action Rules):** 當收到屬於此規則的封包後，該如何處理 (封裝、轉

送、儲存、丟包、...)。

- **BAR (Buffering Action Rules):** 決定對緩衝的處理方法，屬於輔助性規則。
- **QER (QoS Enforcement Rules):** 定義所要應用的 QoS 規定，屬於輔助性規則。
- **URR (Usage Reporting Rules):** 該如何計算與回報流量資訊，屬於輔助性規則。

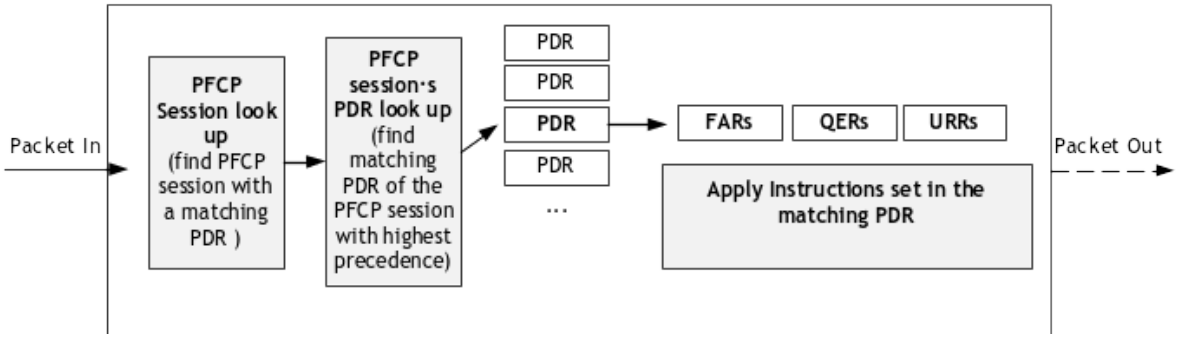


圖 2.3: UP 封包處理流程 (取自 TS 29.244 Figure 5.2.1-1 [5])

圖 2.3 顯示出了 UPF 對於封包的處理過程，當用户端封包進入後，UPF 會從 PFCP 連線中查找出符合封包的 PDR (PDR 是 UPF 內唯一的)，當查找到符合條件的 PDR 後會從 PDR 內儲存的指定 FAR/QER/URR ID 查找相對應的 FAR/QER/URR，來決定針對這個封包相對應的轉送/服務品質/流量計算回報規則，又如果是 FAR 且規則是將封包存入緩衝區，則 FAR 內部可能儲存指定 BAR ID 以查找 BAR 並以此規則對緩衝作出處理方法。

另外在核心網路內部的用户端則是延用 4G 時期的 GTP (GPRS Tunneling Protocol) 協定來作爲封包的穿隧協議，除了 N6 介面 (UPF 與 DN 的中間介面) 外，核網內的用户端介面包含 N3 (RAN 與 UPF) 與 N9 (UPF 與 UPF) 都是使用 GTP 協定相連。GTP 協定在封裝時會於網路層 (network layer, l3) 之下再包覆一層 GTP 標頭與新的 IP 標頭，而在有 GTP 標頭的狀況下，UPF 會透過查詢 GTP 內的 TEID 取代查詢傳統 IP 作爲識別並作出相對應的轉送動作。而整個用户端封包流程從手機到網際網路會是：封包從 UE 送到 RAN 後被 RAN 封裝成 GTP 封包並打上核網分配的 TEID，之後封包被送到 UPF，UPF 查看 TEID 並透過 PDR 與 FAR 的規則查找與應用，把 GTP 解封裝後送到 DN，完整協定疊 (protocol stack) 如圖 2.4。

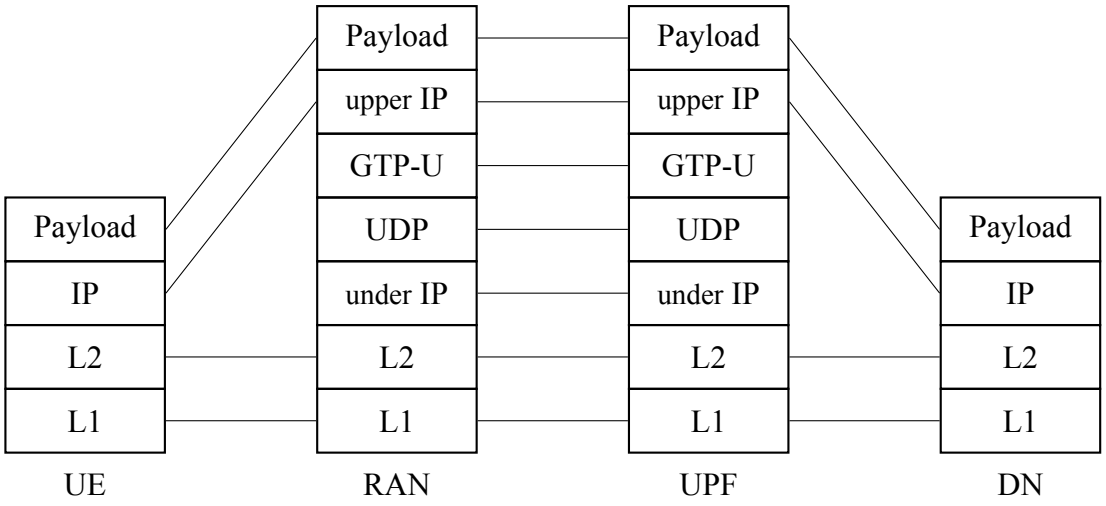


圖 2.4: 用户端協定疊

2.1.3 5G 常見流程

在核心網路中有定義了大量不同的流程 [6] 來適應不同的情境，而其中我們認爲有主要四個流程是在核心網路中最重要也是最長觸發的，對這四個流程的觀察可以讓我們對核心網路有更快速的瞭解，同時因爲其重要性，也適合作爲實驗評估時的基準。

- **Registration:** 註冊是 UE 開啓後第一個進行的流程，UE 需要透過註冊流程才能跟核心網路索取認證 (authorized)、服務 (services)、移動追蹤 (mobility tracking)、與可達性 (reachability) 等功能。另外註冊並不僅限於用户設備連接至核心網路的那一刻，在用户與核心網路的連接期間，依舊會根據不同的場景執行 Registration，主要分爲 Initial Registration，在 UE 開啓電源後嘗試連接至核心網路時所使用；Periodic Registration，當 UE 處於 CM-IDLE 的狀態時，定期與核心網路回報自身存在；Mobility Registration，當 UE 離開 Registration Area 到達新的 Tracking Area 後，更新 UE 狀態時所使用；Emergency Registration，當 UE 僅試圖使用核網所提供的緊急服務時所使用。由此可知，只要 UE 試圖與核心網路連結，必定會執行該流程，可見 Registration 在核心網路中的重要性。
- **Session Management:** 5G 核心網路所提供給 UE 的其中一個重要服務即是與資料網路 (Data Network) 的連接，資料網路並不僅限於網際網路 (Internet) 也包含了 IP 多媒體子系統 (IMS) 亦或是私有網路。UE 爲了連線至資料網路，將會發起 PDU Session 的建立請求，流程完成後，會建立起 UE 和資料網路之間的用户平面，才能進行資料交換。同時在 UE 進行移動時，也可能因爲更換 RAN 或甚至 UPF 而需要經常性的作連

線的修改 (session modification)。如果沒有進行這個流程，UE 將無法連線至網路以獲取所需的資源。

- **Handover:** 換手 (Handover) 爲核心網路支援使用者移動性質的關鍵，同時確保服務的連續性。UE 在行動網路供應商提供之服務範圍中移動時，由於從正在提供服務的基地臺 (base station, RAN) 所接受訊號變差，UE 爲了避免服務品質低落，必須從原先提供服務之基地臺切換至其他訊號較強之基地臺從而維持相同服務品質，同時必須讓使用者不因切換時所造成的中斷延遲時間而感受到使用體驗上的低落。因此換手時間將直接影響了使用者體驗，並且隨著 UE 於長距離下的高速移動，將會大量觸發換手機制，此情形下，確保換手時的低延遲將顯得更爲重要。
- **Paging:** Paging 爲核心網路用來尋找 IDLE 狀態中的 UE 並且觸發訊號連接的過程，由於移動設備需要考慮電源管理 (power management)，在不需要使用到服務時移動設備會進入 IDEL 狀態來關閉連線以減少電源消耗。而在 UE 進入 IDLE 狀態後，若資料網路需要傳送給 UE 的下行封包抵達核心網路，就需要透過 Paging 尋找 UE 接著觸發 Service Request 建立下行 PDU Session。因此低延遲的 paging 流程表示能夠讓 UE 即時的接收到下行封包，也降低了 UPF 儲存大量下行封包下的負擔。同時若沒有 paging 流程，UE 將會需要時時保持與核心網路的連線而可能導致大量的電源消耗。

2.2 現存 5G 核心網路專案

網路上有一些現存的核心網路專案，有一些是開源的另一些則否，像是 NextEPC [7]、Open5GS [8]、OpenAir-CN [9] 等等，但現行的核心網路專案較少看到符合 3GPP Release 15 (R15) [10] 的專案，而 free5GC 是其中少數符合 3GPP R15 定義的 5G 核心網路開源專案，透過符合 R15 的核網，我們才能完整測試 3GPP 定義的 5G 功能、接入 5G RAN 作測試與驗證，而唯有使用開源專案，我們才能快速的開發與修改我們所需要的功能、驗證我們的想法，而不需要自己重新完整復刻一個核心網路。下面我們介紹一些現存較知名的核心網路專案：

專案	4G/5G	開源	描述

NextEPC [7]	4G	v	網路上以知最完整的 4G 核心網路開源專案，幾乎包含所有 EPC 之功能如 MME、S/P-GW、HSS、PCRF 等，但無 5G 功能。此組織後來開發了 Open5GS 來實作 5G 核心網路。
Open Air Interface [11]	4G	v	OpenAirInterface (OAI) 是一個專注於 4G RAN 的開源專案，爲了驗證其 RAN 之功能，OAI 實作了 OpenAir CN [9] 核心網路，但其核心網路僅有部分功能用以驗證 RAN 端，而無實作核網之大部分主要功能。此組織後來有開發 5G 之 RAN 與驗證用 5G 核心網路 OPENAIR-CN-5G [12]，但根據其網頁 [13] 描述，其核心網路目前僅開源了 AMF、SMF、與 NRF，並無完整網路功能。
Internship -5GCN [14]	5G	v	實作 5G 控制端基於 RESTful SBI 的網路功能，目前僅有 AMF、NRF、SMF、UDM，並未完整實作所有必要功能。
Open5GS [8]	5G	v	基於 NextEPC 發展出的 5G 核心網路，具備大部分 5G R16 所需之網路功能，目前主力開發於 5G NSA 架構，可向下相容 4G EPC，然對於 5G SA 之功能尚不完整。
Metaswitch Fusion Core [15]	5G	x	Metaswitch 是一個基於雲端服務的高效能高可靠度的 5G 核心網路，其支援包含 4G LTE 與 5G NR 的 RAN 接入，其中它更可以支援可程式化的用户端服務，然而 metaswitch 目前並不提供開源讓開發者使用。
free5GC [16]	5G	v	第一個 5G SA 架構的開源專案，目前有實作大部分 5G R15 網路功能，並且有額外實作 non-3gpp 的功能，然而不支援 NSA 架構，並且在可靠性與效能上目前並不完善。

OpenUPF [17]	5G	v	基於 3GPP R16 所開發出來的 UPF，底層使用 DPDK 提高高效能轉發，並且可以部屬於 K8S 上提供可擴展性，然而並無實作控制端，僅有用户端 UPF 功能。
UPF P4 PoC [18]	5G	v	驗證基於 P4 (Programming Protocol-Independent Packet Processors) 實作出 UPF 的概念，目前尚在 PoC 階段並且僅有實作 UPF 部分。
upg-vpp [19]	5G	v	基於 3GPP TS 23.214 [20] 與 TS 29.244 [5] 所實作的 GTP-U 閘道器 (gateway)，底層使用 Cisco 開發的 Vector Packet Processor (VPP) [21]，僅實作 UPF 之功能。
upf-xdp [22]	5G	v	基於 Linux Express Data Path (XDP) 所實作之 UPF 的概念性驗證，目前停留於驗證階段且僅實作 UPF 部分。

表 2.2: 5G 專案介紹 [23]

2.3 現行方案的挑戰

5G 核心網路透過改變架構希望達到三大訴求：增強行動頻寬 (eMBB)、高可靠低延遲 (URLLC)、大量機器連線 (MMTC)，以達到更廣泛的使用體驗，例如 VR 應用便要求低於 20ms 的延遲 [24]，而 AR 更是需要低於 5ms 的延遲才能有最佳的使用者體驗 [25]，另外像 4K 影像串流就至少需要 32Mbps 的下載頻寬 [26]。

然而就目前的 5G 架構，儘管在 RAN 端透過 mm-Wave 已經可以把延遲降低至 1ms [27]，並將頻寬提升至 10Gbps，但核心網路尚不能保證提供高流量低延遲的服務，其原因包含 1. 多個 NF 與 SBA 的設計，由於 5G 的連線建立流程與 4G 大致相同，又將不同功能細分成不同 NF，各 NF 間的溝通勢必會增加整個流程的延遲，2. NFV 的架構將使控制訊號無法依賴硬體加速，反而相較於 4G 以前的硬體網路功能更爲高延遲低流量，3. 流程保持 4G 的階段性完成，在控制端完成功能前用户端都無法取用資源，這讓

控制端大幅影響用户端的使用效能。例如取得需要先完成註冊與連線建立流程，而此論文 [28] 指出在 4G 時，光是 UE 的連線建立流程就可能會消耗 999.5ms，而在高速移動下更是需要完成多次換手 (handover) 流程，但因換手需消耗 1.9 秒 [28]，導致高速移動時的行動網路體驗將會大打折扣，4. 於 2023 年會有 44 億個 M2M 或 IoT 裝置連線上行動網路 [29]，如此大量的裝置勢必無法被現有的核網資源所承載。

以下我們列舉了現行核心網路所遇到的挑戰：

- **挑戰 1. 控制層處理流程:** 由於 5G 核網 SBA 的架構，相較於 4G 核網，控制訊息交換的數量大量提升，這使的控制訊息的傳遞速度延遲被等倍放大，光是註冊流程在核網內就至少有超過 19 個控制訊息，這還不包含每次向 NRF 進行的 NF 查找訊息以及例外處理訊息；另外爲了要提供更多網路服務，控制訊息內含的資訊元素 (IE) 也越來越多，如果可以加速資訊元素序列化、HTTP2/RESTful 結構的建立、乃至於 TCP 的傳輸效率，便可能大大提升控制層效能，進而讓用户層更快的被處理到。
- **挑戰 2. 用户層規則查找:** 在用户端方面，UPF 會對進入的封包進行規則的查找作爲轉發的依據，然而相較於 4G 時只需要對 TEID 或 IP 進行分析，5G 的規則的越趨複雜，包含五種規則的查找、優先序的比較，更甚者隨著 UL-CL (Uplink Classifier) 等功能的加入，會有像 SDF (Service Data Flow) 過濾器的出現，需要進一步對 5 tuples 作過濾，如果可以加速規則查找速度便有機會大幅提升用户端的效能。
- **挑戰 3. 換手時複雜的下行封包處理:** 根據 TS 23.502 的換手流程 (見圖 2.5 流程 3a 與 3b)，在換手時爲了讓下行流量不間斷，5G 網路會在 3a 或 3b 流程上進行直接或間接轉送，亦即此時封包會經過 S-UPF 送到 S-RAN，如果是直接轉送會透過 Xn 介面從 S-RAN 將下行轉送到 T-RAN；如果是間接轉送會將封包先回送到 T-UPF，再送到 T-RAN。可是這樣的過程大量消耗流量並增長下行路徑，尤其是當換手的流程過於緩慢，更是可能造成行動網路移動時的連接不順。

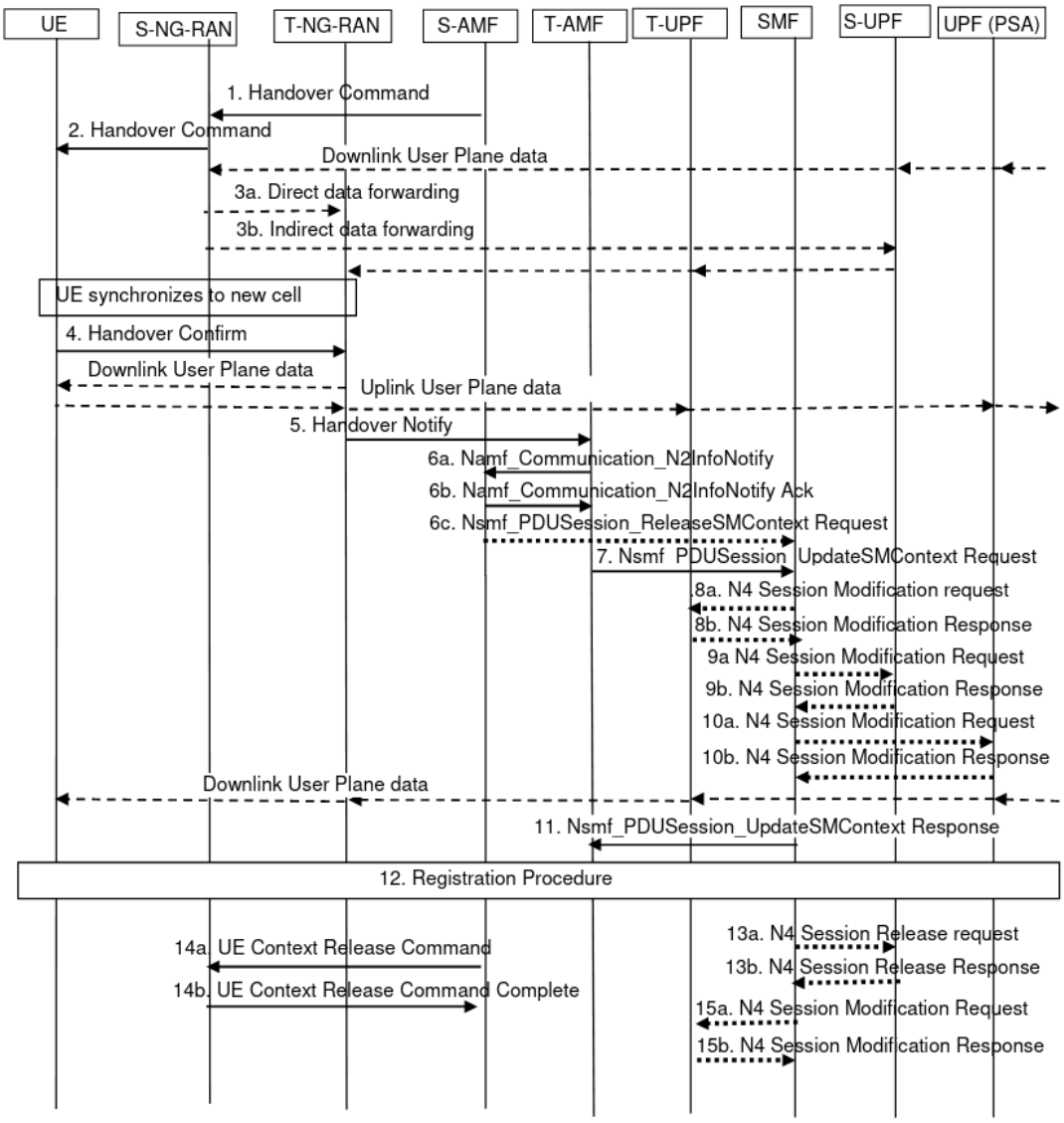


圖 2.5: N2 Handover 執行階段流程 (取自 TS 23.502 Figure 4.9.1.3.3-1 [6])

三、 系統架構與實作

3.1 實驗平臺: free5GC 系統介紹

free5GC 是一個現行的開源 5G 核心網路方案，在版本 v3.0.4 上原生支援 5G SA (Standalone) 架構，也就是不需要藉由 4G 網路作爲中間層，包含核心網路與無線訊號接入端 (RAN, radio access network) 都可以並且必須是純 5G 之架構。而 free5GC 在 v3.0.4 上已經有實作 AMF、AUSF、NRF、NSSF、PCF、SMF、UDM、UDR、與 UPF (見圖 3.1)，對於非 3GPP 接入 (non-3gpp access)，free5GC 亦有 N3IWF 的支援，是目前開源 5G 核心網路方案中，架構上最爲完善的軟體。由於其開源的性質與廣大的設群討論，基於 free5GC 進行修改開發無疑是最爲方便有效之選擇。

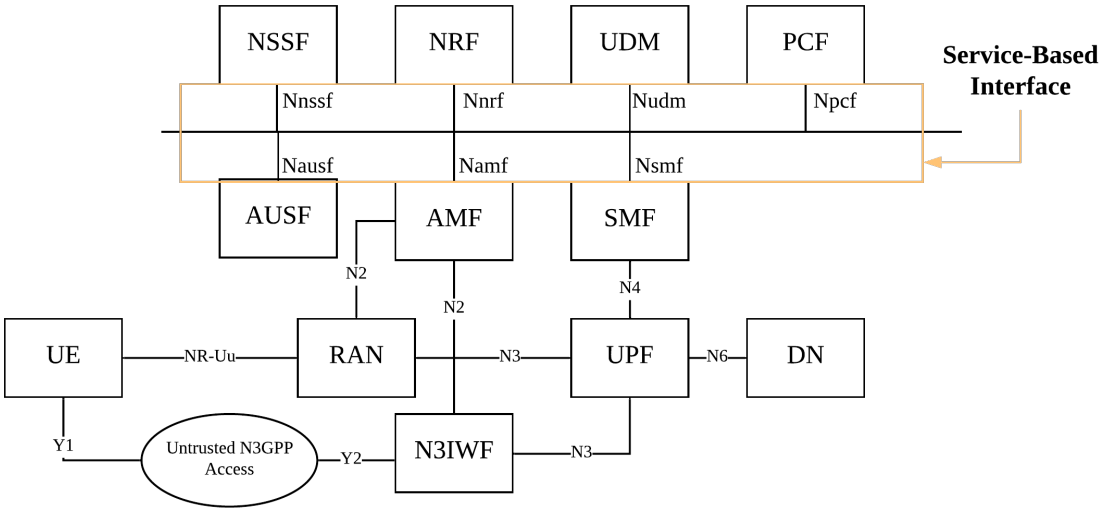


圖 3.1: free5GC 架構 [16]

free5GC 源自於 NextEPC [7]，一個 4G 開源核心網路，於版本 v1.0.0 時將 NextEPC 之 MME、S/P-GW 等運算、管理連線之 NF，改爲 5G 之 AMF、SMF、與 UPF，並保留了 4G 的 HSS 與 PCRF 作認證之功能，此階段是對於 5G 架構的概念性驗證 (PoC, Proof of Concept)，並沒有完全按照 3GPP 的標準實作。並且由於 NextEPC 是使用 C 語言開發，在此版本下，free5GC 亦是使用 C 語言開發，並且大量繼承使用 NextEPC 之功能與函式。

於版本 v2.0.0 時，free5GC 完全重構了核心網路，捨棄掉 NextEPC 的程式碼而改爲重新開發一個基於 3GPP R15.3 的 5G 核心網路。由於考量到 Go 語言有許多適合撰寫網路程式的特性，諸如原生提供大量方便使用之標準函式庫 (standard library)、內建垃圾回收機制 (GC, garbage collection)、可快速引用第三方函式庫 (third party library) 等等，而選擇使用 Go 語言作來撰寫所有控制端 NF，唯獨 UPF 因爲考量到效能以及未來可能需要與其他高效能函式庫間接的關係，而沿用 C 語言開發。

經過了對 free5GC 的程式碼的研讀後，我們瞭解 free5GC 大致上把控制端的 NF 大致規劃成三種部件：SBI、中心處理部件、與其他。以 SMF 爲例 (見圖 3.2)，分爲 SBI、Flow Procedure、與 PFCP。SBI [30] 全名爲 service base interface，是 3GPP 於 R15 設計出來做爲控制端溝通的協定，主要基於 HTTP2、RESTful API 設計，分爲 Consumer 與 Producer，由 Consumer 向 Producer 要求 (request) 資訊。相較於 HTTP，HTTP2 提供了更少的網路延遲，而使用 RESTful API 則可讓 API 有更佳的結構，並且 3GPP 直接採用 OpenAPI 格式來定義 RESTful 的格式，得以在設計好 OpenAPI 文件後，可讓人類快速讀懂，也可以透過 OpenAPI 工具直接產生出相對應的程式碼，而 free5GC 正是透過 OpenAPI generator 工具把 3GPP 定義的 SBI YAML 檔案轉譯出 Go 語言的 gin server 程式。PFCP [5] 是 SMF 獨有的協議，不存在於除 SMF 與 UPF 外的其他 NF，主要是用來讓 SMF 與 UPF 溝通，並且對 UPF 下達相對應的封包轉發指令，於 3GPP 在 R14 時爲了設計 4G EPC 中 CUPS (Control and User Plane Separation, 控制與用户端分離) 而採用的。PFCP 建立於 UDP 之上並採用 TLV (type, length, value) 的格式設計，封包內容裡每個 IE (information element) 都會有三個主要成分：2 byte 的 type 用來描述 IE 種類、2 byte 的 length 用來描述 IE value 的長度、以及 length byte 的 value 用來敘述 IE 的內容。而 TLV 亦可巢狀使用，即一個 TLV 的 value 內可以包含其他一個或多個 TLV 使用。透過 TLV 的設計，PFCP 可以將訊息有效的壓縮長度。而 flow procedure 則是負責整個 SMF 的邏輯運算，經過 PFCP 或 SBI 解碼過的內容會被統一傳送給 Flow Procedure 並且由 Flow Procedure 來決定相對要做的事情。

其他控制端 NF 架構上大致類似，若是只有 SBI 介面的 NF 像是 UDM、PCF 等等，會捨棄其他部件，而像是 AMF 有 SBI 與 NGAP，則會用 NGAP 取代其他部件。

至於 free5GC 用户端的 NF UPF 則是主要分成三個部分：N4 Handler、UPDK、與 gtp5g [31] (見圖 3.3)，N4 Handler 主要是負責處理從 SMF 送下來的 PFCP 控制訊號，他

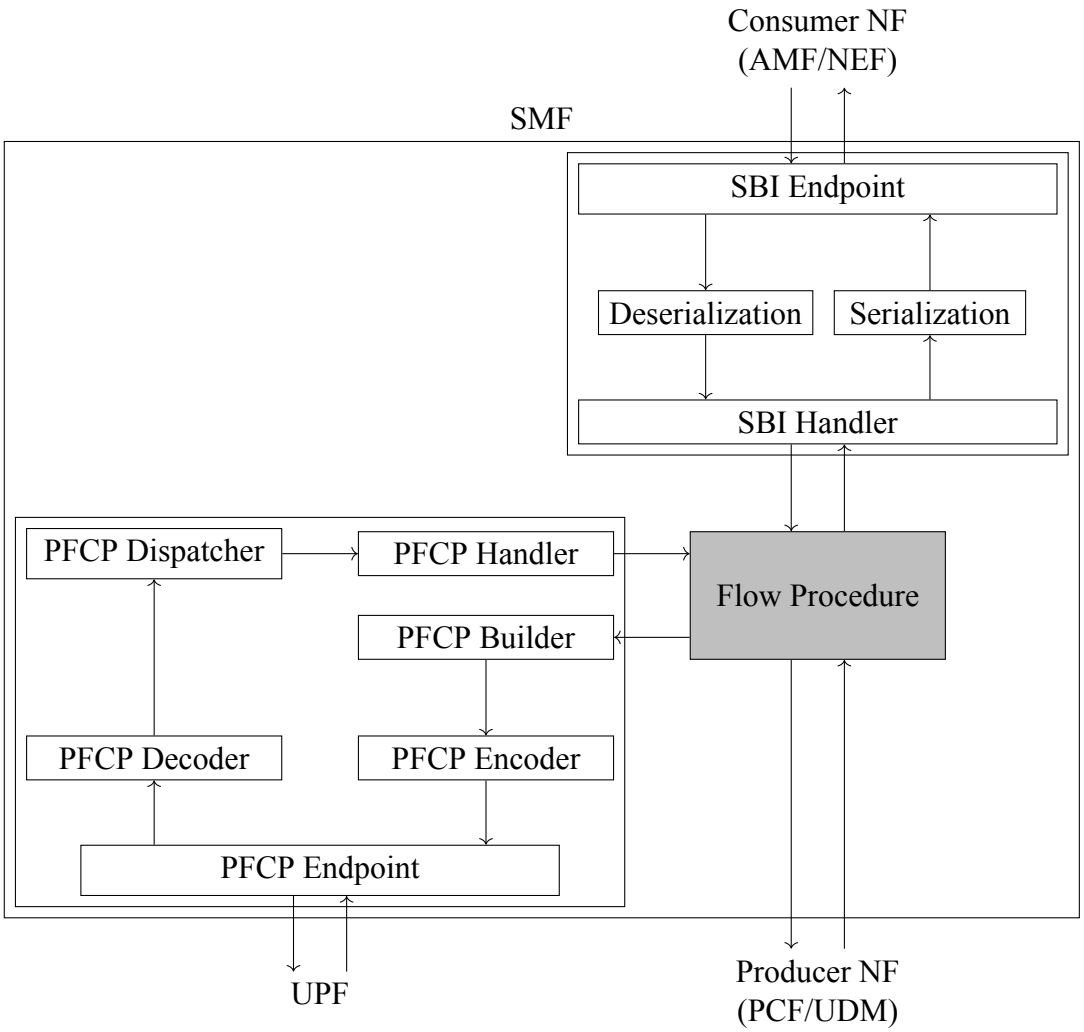


圖 3.2: free5GC 原生之 SMF 架構

會直接透過 Linux UDP Socket 獲取 N4 訊號，解碼 PFCP 後翻譯成 UPF 看得懂的格式，比較有趣的是，因爲 PFCP 是遵守 TLV 格式的，因此 free5GC 在撰寫這部分時，是直接解析 3GPP 文件後透過自己撰寫的程式碼生成器 (code generator) 來產生 encoder 與 decoder。UPDK 主要是負責處理用户端訊息的邏輯，向上他會接收到 N4 Handler 所下的 PDR (Packet Detection Rules) / FAR (Forwarding Action Rules) 指令儲存起來，並在收到相對應的 GTP (GPRS Tunneling Protocol) 封包後協助 gtp5g kernel module 進行相對應的封包處理。而 gtp5g 則是負責協助處理用户端介面 N3、N6、與 N9 的 kernel module，在 N6 介面上，會透過 netlink 創造一個虛擬裝置 (virtual device) 來接收從 DN 來的封包，N3 與 N9 則是直接從 netlink module 拿到 routing 進來的封包。

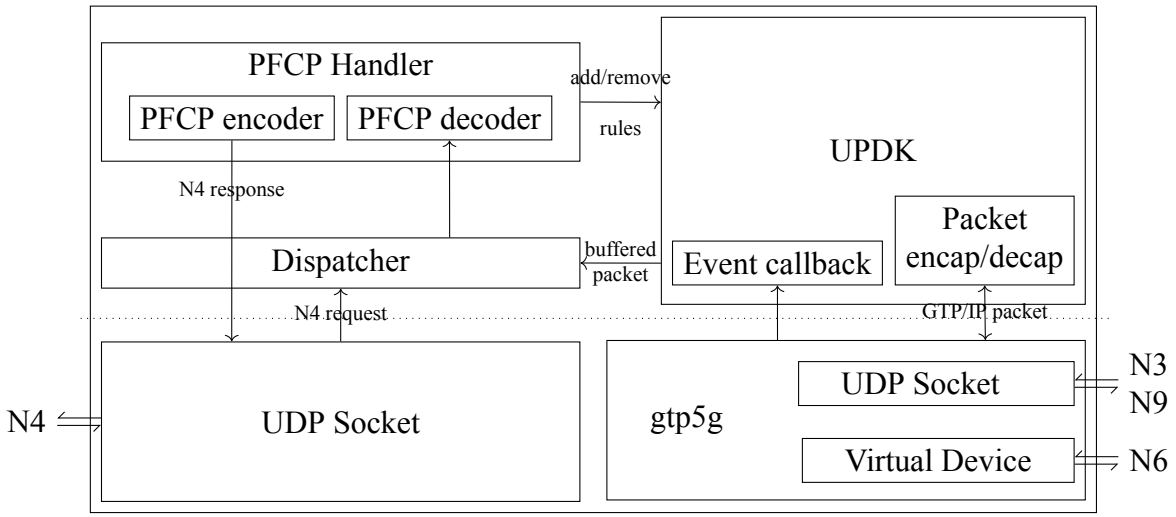


圖 3.3: free5GC 原生之 UPF 架構

3.2 系統層 API: OpenNetVM 系統介紹

OpenNetVM [32] 是一個由加利福尼亞大學河濱分校 (UCR) 與喬治華盛頓大學 (GW) 共同開發，基於 DPDK、Docker 容器、與共享記憶體 (shared memory) 設計的開發平臺，目標在於提供 Network Function 有更好的管理與效能，有幾個主要的優勢：

- **Container-based NFs:** OpenNetVM 平臺透過 Docker 容器管理運行於平臺上的 Network Function，因此只需要用一般 user space 程式的撰寫方式開發網路功能便可以輕易部屬至 OpenNetVM 平臺之上。
- **NF Manager:** OpenNetVM 平臺透過管理程式 NF Manager 可以持續追蹤平臺上 Network Function 的狀態，並且分配相對應的封包給予 Network Function 內。

- **SDN-enabled:** NF Manager 透過 Open Flow 協議向上支援與 SDN 控制器溝通，並可以讓 SDN 控制器組合多個 Network Function 建立服務鏈 (service chain)。
- **Zero-Copy IO:** 封包透過直接記憶體存取導入 NF Manager 可以存取的共享記憶體空間中，再由 NF Manager 分配給 Network Function，透過直接存取封包的能力，過程中無須經過多於的封包複製。
- **No Interrupts:** 透過 DPDK 的 polling 模式，封包直接由特定 CPU 固定拉取，而無須透過傳統的中斷 (Interrupt) 處理的方式處理，這可以讓系統在 10Gbps 甚至更高的流量時都還保持著線性的封包處理效能。
- **Scalable:** Network Function 可以被輕鬆的複製擴充，NF Manager 會自動對封包做負載平衡 (load balance) 以提供最大網路效能。

根據作者描述 [33] [34]，OpenNetVM 的架構主要分成三個部分：NF Manager、NFlib 與 Network Functions (如圖 3.4)。NF Manager 負責與網卡溝通、維護 flow table、管理 NF 間的通訊，整個 NF Manager 包含三個線程 (thread)：Manager thread 負責管理共享記憶體跟追蹤 NF 狀態、RX thread 負責從網卡收取封包並分配給對應的 NF、TX thread 負責轉送 NF 要送給其他 NF 或要送到網卡的封包。NFlib 是 OpenNetVM 提供給使用者的 API，透過 NFlib 使用者的程式得以與 NF Manager 溝通以註冊 NF 狀態、獲取封包等。而 NF 則是指使用者開發的程式，NF 會被放置於 Docker 容器內並且透過 NFlib 與 NF Manager 溝通、讓 NF Manager 管理，每個 NF 在啓動時都會被分配到一個 receive ring 與 transmit ring，NF 會透過 NFlib 向 receive ring 獲取進入封包，並將要送出的封包放入 transmit ring 中。

在 OpenNetVM 平臺被啓動時，NF Manager 會首先被啓動，並且直至 OpenNetVM 平臺關閉前都會持續運作，NF Manager 會透過 DPDK 在 hugepage 內分配出一塊共享記憶體，用以儲存與維護所有封包，並且此記憶體內的虛擬位址會共享於可以存取此封包的 NF 而無須做額外的轉換或複製。當主機網卡收到封包後，RX thread 會利用 DPDK 將封包輪詢至共享記憶體內，由於 DPDK 的 DMA 與核心旁路 (kernel bypass) 機制，得以讓封包享有零複製 (zero-copy) 的效率，RX thread 把封包輪詢至共享記憶體後，會逐一檢查封包並將包含封包位址與元資訊 (metadata) 的 packet descriptor 轉送給相對應 NF 的 receive ring，透過這 packet descriptor，NF 無須再複製封包便可存取到封包的位址與內容。而 TX thread 會執行與 RX thread 類似的功能，差別在於 TX thread 是

負責轉送從 NF transmit ring 送出的封包的 packet descriptor，如果要傳送至另一個 NF，TX thread 會將 packet descriptor 提供給另一 NF；而如果是要送出網卡，TX thread 則會透過 DPDK 將封包從網卡送出。

在 OpenNetVM 上執行的 NF 會透過 NFlib 與 NF Manager 溝通，在 NF 開始運行時須向 Manager 註冊服務，並且會提供 NFlib 回呼函式 (callback function)，當有 packet descriptor 送進 NF 的 receive ring 時，NFlib 會呼叫回呼函式讓 NF 得以做出相對應的操作，而當 NF 決定好操作後，便會將操作與相對應的 packet descriptor 放進 transmit ring 讓 Manager 執行。另外值得注意的是，除了 NF Manager 中的 Manager thread、RX thread、TX thread 會各佔據一個 CPU 核心外，每個 NF 亦會獨佔一個 CPU 核心。

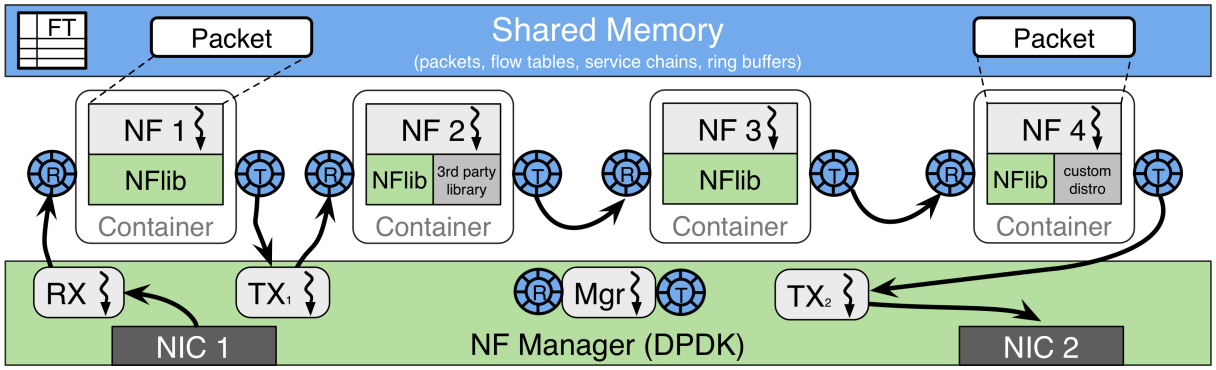


圖 3.4: OpenNetVM 架構 [32]

透過 OpenNetVM 的 API (Application Programming Interface) 設計，我們可以快速開發基於 DPDK 與共享記憶體的網路程式，因此開發 NF 時不需要考慮系統底層實作細節，大大的減少了開發所需成本。另外由於平臺是基於 Docker 容器包裹 NF，因此只需要可以跑在容器內的程式都可以部屬進此平臺內，同時可以享受容器隔離 (isolation) 的安全性，以及快速部屬、擴張的方便性。另外由於有 NF Manager 的幫助，我們無須對 NF 作進一步的管理，只需著重於單一 NF 內部功能的開發即可，大幅減少開發時對對管理與部屬所消耗的心力。

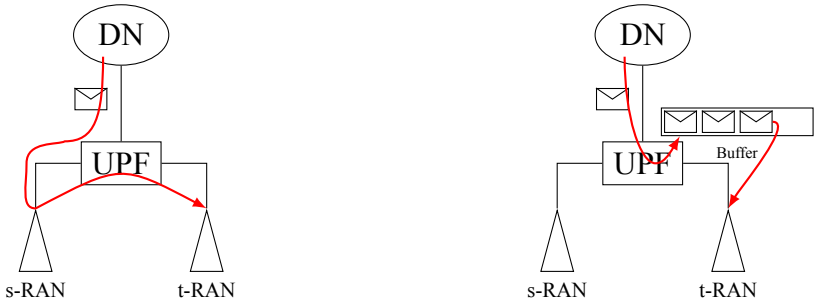
3.3 系統設計目標

我們希望能夠在不改變 3GPP 標準規定的框架下，提升包含核心網路控制層與用户層的效能，減少控制層訊號傳遞的延遲以加速連線建立速度與移動穩定度，並且嘗試透過更好的使用者端規則查找與封包處理方式，得以使使用者流量延遲降低、頻寬增

加。藉由以軟體的方式實作符合 3GPP 標準的 NF，可以依然保有 5G 架構下的彈性、可擴充性與溝通能力。以下列舉我們認爲需要做的增強方式：

- **部屬 NF 於同一節點:** 由於網路功能虛擬化的興起，未來核心網路將不再是每個功能都放置於不同機器上，而是會以軟體或容器的方式呈現，透過此一場景的觀察，我們認爲 5G 核網的部屬有別於傳統 4G 核心網路，將可以將不同 NF 部屬於同一節點以增加訊息溝通之效率。而當我們需要將核心網路擴展時，僅需要透過 OpenNetVM 的 NF Manager 將流量導向不同節點，於額外節點上部屬新的 NF，並用 Network Slice 切割使用者連線，便可在不失溝通效能的前提下，輕鬆達到擴展核心網路之功能。
- **使用共享記憶體溝通:** 透過在同一節點上部屬高頻率溝通的 NF，我們嘗試取用同一節點的好處，利用 IPC 取代 RPC 溝通，同時透過仔細的共享記憶體操作，甚至可以達到無須移動訊息位址便可傳遞控制訊號。利用這樣的改善，我們不但可以減少 HTTP2 序列化 (serialization) 與反序列化 (de-serialization) 的效能流失，甚至可以捨棄 TCP/UDP 封裝與協定交換的過程。
- **越過核心處理:** 傳統 Linux 網路介面 (socket) 需要透過與核心 (kernel) 溝通來獲取或傳送封包至用户空間 (user-space)，但核心處理的過程中會耗費大量的時間與運算資源，同時必須遵守核心的網路疊 (network stack)。透過 Intel DPDK (Data Plane Development Kit) 的操作，我們可以越過核心處理，直接把封包從網卡輪詢至用户空間，大幅減少了經過核心的效能消耗。
- **TSS 規則查找:** 以往使用者端封包進入查找相對應規則時，多是使用線性搜尋，我們修改使用了 Tuple Space Search [35] (TSS) 以增加搜尋速度。TSS 會對規則進行分堆，而在相同堆中的不同規則以雜湊表 (hash table) 儲存，因此在相同堆中的規則僅需複雜度 $O(1)$ 即可查到，又因堆數必小於等於總規則數，透過 TSS 查找的最差複雜度必定小於等於線性搜尋的速度。
- **調整換手流程:** 在換手流程中，爲保持用户端下行流量的連續性，行動網路會透過直接或間接轉發將下行封包從換手前的 RAN 轉發到換手後的 RAN，但這樣的流程將會增加下行路徑距離而增加延遲 (見圖 3.5)。我們透過修改流程，將下行封包緩衝於 UPF 內以減少下行封包行走之路徑距離。
- **減少相同訊息的觸發次數:** 在 NF 溝通的過程中，有可能會發出相同的訊息，其中以 NF 向 NRF 發出 NF Discovery 訊息尤爲明顯，我們透過快取 (cache) 的使用，讓相同

的 NF Discovery 訊息只發出一次以減少控制端溝通時的訊息量，的以降低總體流程的延遲。



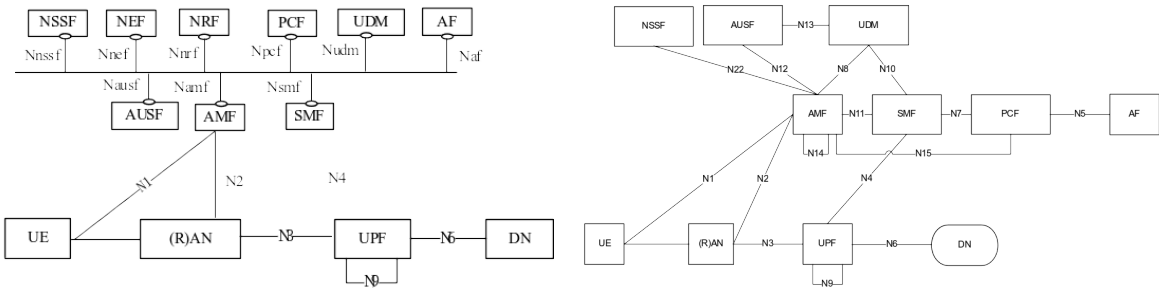
(a) 換手間接轉送　　(b) 換手封包緩衝

圖 3.5: 換手下行封包處理

因應以上系統目標，我們選擇使用 free5GC 與 OpenNetVM，重新設計了 LH5GC 核心網路平臺。

3.4 SMF 移植細節

雖然 5G 使用 SBI 架構整合了大部分的控制端介面，但還是保留了 NGAP 跟 PFCP 於 N2 及 N4 介面 (見圖 3.6)，其中 N2 是介於 RAN 與 AMF 之間的通訊，透過 NGAP 協定，類似於 4G 時代的 S1AP 協定，而 N4 則是 SMF 與 UPF 間的介面，沿用 4G R10 之後 CU 分離概念所使用的 PFCP 協定。在這主要的三個介面中，SBI 是基於 HTTP2，傳輸層 (L4, transport layer) 是使用的是 TCP；NGAP 的傳輸層使用 SCTP；而 PFCP 的傳輸層使用的是 UDP。



(a) 5G 系統架構，控制端用 SBI 表示
(取自 TS 23.501 Figure 4.2.3-1)

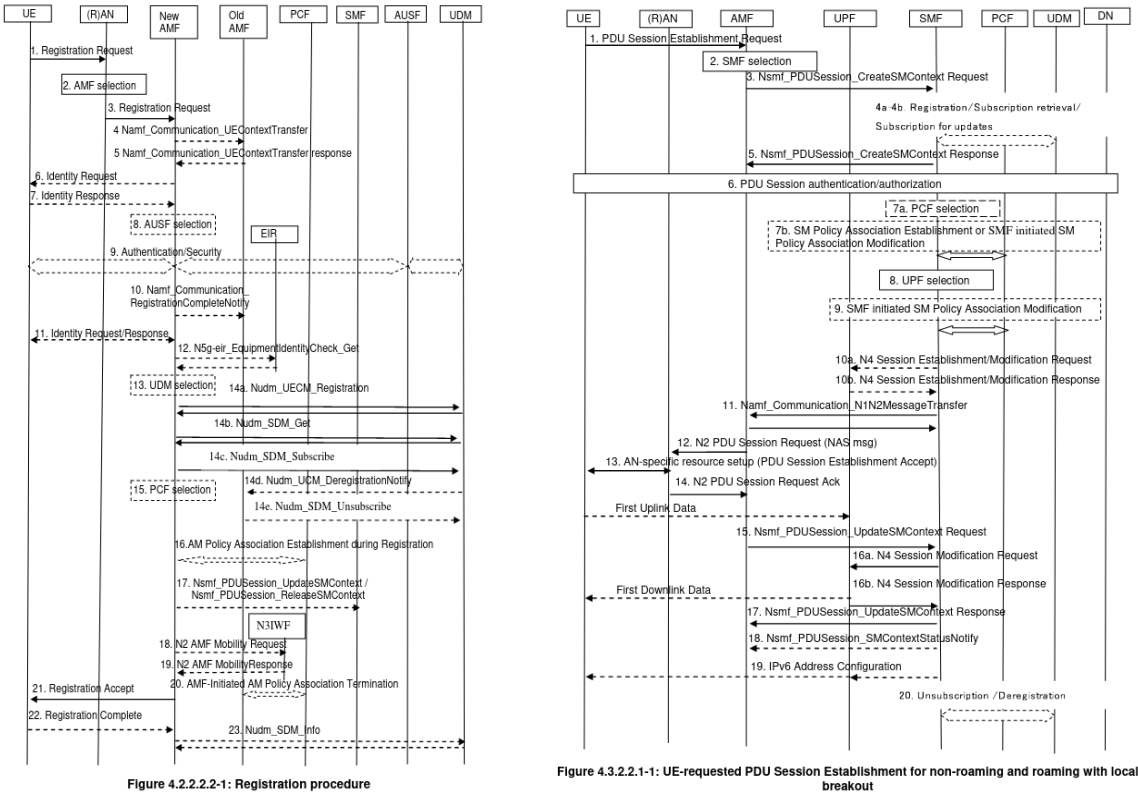
(b) 5G 系統架構，控制端用參考點表示
(取自 TS 23.501 Figure 4.2.3-2)

圖 3.6: 5G 系統架構 [2]

我們希望專注於改善行動網路最常見的連線行爲，UE 連線上網行爲的效能，

而這個情境需要幾乎所有 NF 的參與，但其中有三個 NF 扮演著最重要的角色，AMF、SMF、和 UPF，如同章節 2.1所描述的，AMF 主要負責處理接入 (access) 與移動 (mobility)，SMF 負責連線的管理 (session management)，而 UPF 則是用户端 (user plane) 封包最重要且唯一的轉送處理單元。

連線過程主要可以分爲兩個階段：註冊 (register) 與連線建立 (Session Establishment)。連線過程會做諸如確認 UE 位置、驗證使用者資料、確認使用者能力、確認一些上下文、或給予 ID 等，主要由 AMF 參與並且向其他 NF (ex. UDM, AUSF) 詢問一些資訊。而連線建立主要會幫助 UE 建立正確的連線路徑、設定 QoS 參數、設定緩衝參數等等。在這個階段裡，除了 AMF，SMF 與 UPF 也會在這個階段內參與運算，此時 SMF 負責管理與建立連線路徑：向核心網路端他會通知 UPF 建立路徑，而向 RAN 端他會透過 AMF 的中繼 (relay)，向 RAN 發出指令建立路徑或詢問資訊。



(a) 註冊流程

(取自 TS 23.502 Figure 4.2.2.2.2-1)

(b) 連線建立流程

(取自 TS 23.502 Figure 4.2.2.2.2-2)

圖 3.7: 5G 流程 [6]

爲了加速控制端的控制訊號 (control signal) 可以更快的新增、修改、刪除用户端的會話 (session)，我們希望可以透過加速 AMF、SMF、與 UPF 這三個於 PDU Session 流程中最重要的 Network Function 之間的通訊速度，降低其三者間的通訊延遲 (latency)，

得以在 5G 核心網路進行大量會話的新增、修改、與刪除中，得到最好的效能。

在 AMF、SMF、與 UPF 中，SMF 是負責 Session 管理的 Network Function，即是 Session 管理最重要的部件，因此我們希望優先著重於其控制信號的效能最佳化。而在 SMF 的介面中，又以 N4 介面用以對 UPF 下達 Session 管理的介面最爲重要，因此我們決定於 N4 介面上，透過 OpenNetVM 物理層傳輸效能的優勢取代傳統 Berkeley Socket，藉以提高 N4 介面的傳輸效能，降低控制訊號的延遲。

原生的 free5GC 控制端介面是建構在 Linux 的 Berkeley Socket 上，先將 L4 層以上的負載內容 (payload) 封裝後，再透過呼叫 Berkeley Socket API，來達成封包傳送的功能。但由於 Berkeley Socket 屬於 user space library API，並且在傳送過程中，至少會經過兩次記憶體複製 (memory copy) (會先複製到 per process 的 kernel space memory，再複製到 network device 的 network queue) (見圖 3.8)，相對於後來出現的技術諸如 DPDK 或 XDP，實屬效能不彰。另外，即便是將 Network Function 部屬於同一機器上，由於 Berkeley Socket 屬於 RPC，傳送路徑必定會經過 Network Device，無法自動判定爲同一節點通訊而使用更高效率的傳輸方式。基於以上理由，我們希望透過 OpenNetVM 平臺的性質，同時可以於 RPC 下使用 DPDK 功能，與 IPC 使用 shared memory 功能，實現完全零記憶體複製 (zero copy)，以及可於同一主機下使用 IPC 來達到更佳的傳輸效率。另外除了記憶體複製的問題之外，由於透過 system call 到 kernel 的設計，也會遇到所有 system call 會遇到的問題，像是會有至少兩種中斷 (interrupt)，一個是在呼叫 system call 的中斷，另一個則是因爲網卡與記憶體是透過 DMA (direct memory access) 的關係，會在 DMA 機制上需要在把封包搬進記憶體後由網卡打入中斷來呼叫中央處理器進行下一步處理。同時 kernel call 也會因爲需要上下文切換 (context switch) 到中斷上下文 (interrupt context) 或甚至是遇到程序排程 (process scheduling) 的上下文切換而消耗大量運算資源與效能。

不管是控制端介面上的 NGAP、SBI、或 PFCP 都是建立在 Berkely Socket 上，而如章節 2.3所述，我們認爲 5G 核心網路的 NF 因爲網路功能虛擬化 (NFV)，因此可以放置於同一主機上。除了 NGAP 是 N2 介面的協定，即基地臺與核心網路的連結，因爲基地臺並未虛擬化而是需要部屬在邊緣端。因此我們認爲包含 SBI 與 PFCP 的協定若可以用更高效率的共享記憶體設計取代 Berkeley Socket，將能大幅提高核心網路控制端的效能。更者，由於 SBI 是建立於 TCP 協定之上，還會遇到諸如 TCP 三方交握或流量控制

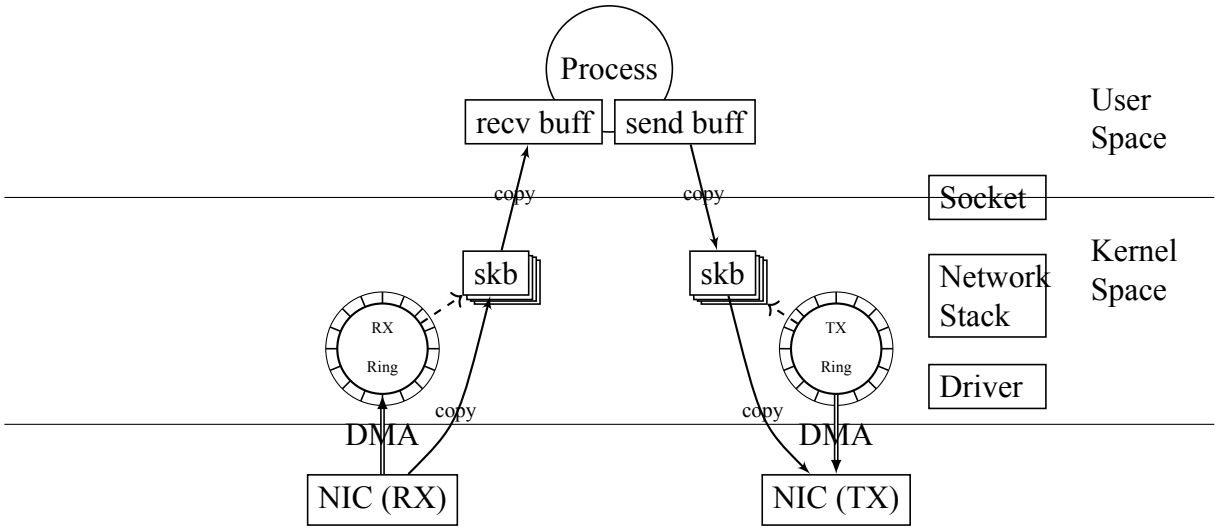


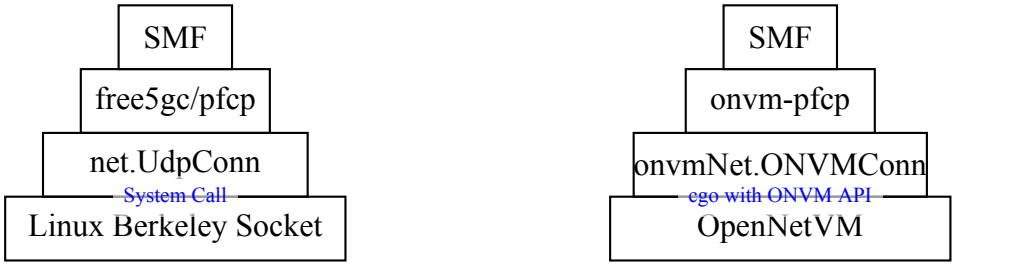
圖 3.8: 核心封包處理架構

(flow control)、擁塞控制 (congestion control) 等機制影響效能，移植至共享記憶體應該能大幅提升效能。

在 SMF 移植的設計上，由於 OpenNetVM 所提供的是 C 語言的 API，而 free5GC 的 SMF 所使用的是 Golang 語言開發，因此需要考慮到如何跨語言移植。在設計之初有考慮是否需要透過 IPC 溝通，另外啓動一個 relay 將 SMF 的訊息 (message) 先透過 IPC 傳送到額外 relay process，再由 relay process 呼叫 OpenNetVM API。但由於發現 Golang 本身有提供 CGO 的功能，透過指定語法可以直接呼叫 C 語言 API，因此我們決定以此方法，減少 IPC 溝通的效能損耗。CGO 提供 C 呼叫 Go 語言函式或是 Go 呼叫 C 語言函式，若是以 C 呼叫 Golang，需要先將 Go 函式庫編譯成 share object library (.so 或.dll 檔)，之後再讓 C 語言透過 share object 的方式呼叫，呼叫過程中由於已經編譯成 object file 並使用 dynamic linking 的方式，是由程式跑起來後 loader 來處理，因此對 caller (C 語言程式) 幾乎沒有任何限制。反之如果是 Golang 呼叫 C 語言函式庫，則會有較多限制，例如 callee 不可以有信號 (signal) 呼叫 (其歸因於 golang 本身訊號覆蓋 (signal mask) 的設計)、部分型別需要做強型別轉換、需要透過 cgo flag 讓編譯器或連結器得到相對 library 絕對位置等等。

儘管透過 C program 呼叫 Golang shared library 在實踐成本上遠低於使用 Golang 呼叫 C library，但我們希望在程式設計上可以提供更高的擴充性 (scalability) 與更加一般化 (generalize)，且若可以設計成函式庫 (library) 形式，會更加利於使用者使用且可以不僅限此專案使用，因此我們希望嘗試以可以直接使用之 Golang 函式庫方向開發。

由於從 OpenNetVM API 上獲取的封包是直接傳回封包所在記憶體位置的指標，又封包爲連結層 (link layer) 之上的內容，因此我們決定開發成類似 Golang 原生之 net 函式庫，透過 Golang 所提供的 interface 性質，直接取代掉 PFCP 底層之 `net.UdpConn` interface，我們命名之 `onvmNet.ONVMConn` interface [36] (見圖 3.9)。



(a) 原生之 free5GC SMF 架構　　(b) OpenNetVM 移植後之 SMF 架構

圖 3.9: SMF 相關性依賴

在設計此函式庫時有部分條件需要考慮，首先，由於 Golang 在透過 CGO 呼叫 C 函式時，有不可呼叫內部有訊號註冊 (signal handling) 的函式，其原因爲 Golang 本身有對訊號作內部取代處理，因此如果透過外部語言另外註冊訊號，會影響 Golang 程序底層訊號處理，但由於 OpenNetVM 在初始化階段需要透過 OpenNetVM 的 API 對系統作訊號註冊，因此會出現問題。而我們參考了網路上的解決方法，將訊號註冊放置 Golang 的 `init` 函式，`init` 函式是 Golang 設計給予在主函式 (main) 執行起來前，預先執行的函式，類似 C 語言的 `__attribute__ ((constructor))`。

然而，使用 `init` 函式會遇到初始化參數無法透過程式執行過程中傳遞，因此我們採用 OpenNetVM 提供的 NF config 設計，透過 config 檔預先傳入 OpenNetVM 平臺之必要參數，諸如 DPDK 參數、Service ID、Instence ID 等。又因爲 OpenNetVM 是使用 Service ID 來決定封包目的地應該傳送至那一個 Network Function，而非使用傳統的 IP 來做 routing，因此我們也對應設計了 ipid.yaml 的檔案預先定義 IP 與 Service ID 的對應關係。透過這樣的設計，函式呼叫者 (caller) 可以直接沿用傳統 TCP 或 UDP 的呼叫，進入 onvmNet 函式之後才會透過此 ipid.yaml 檔案映射到相對應的 Service ID。

另外在方法 (method) 的設計上，因爲我們需要繼承 `net.Conn` 這個 Interface，因此我們實作了此 Interface 所有提供之方法，讓使用者得以直接呼叫。在設計封包收取 (`ReceiveFrom`) 方法時，需要判斷此封包屬於那一個連線 (connection) 的，因此我們設計了 `channelMap` 資料結構，會使每個連線在聽到 (listen) 連線時會開出一個對應的 `channel` 並放入 `channelMap`，當處理者 (handler) 收到 OpenNetVM 給予的封包後，得以

透過 HashMap 的方式以 $O(1)$ 的速度搜尋出相對應的 `channel`，再將封包送入相對應 `channel` 以送到正確的連線中。

設計完 onvmNet 函式庫後，需將 free5GC 內部 PFCP 函式庫內有呼叫到 `net.UDPConn` 的部分改成呼叫 `onvmNet.ONVMConn`。另外由於 free5GC 是透過 go module 來維護套件，因此還需更新 go.mod 之內容。而若有其他 Golang 專案想要嘗試使用 OpenNetVM 平臺，僅需作相同的取代，即可快速移植 (見圖 3.9 (b))。

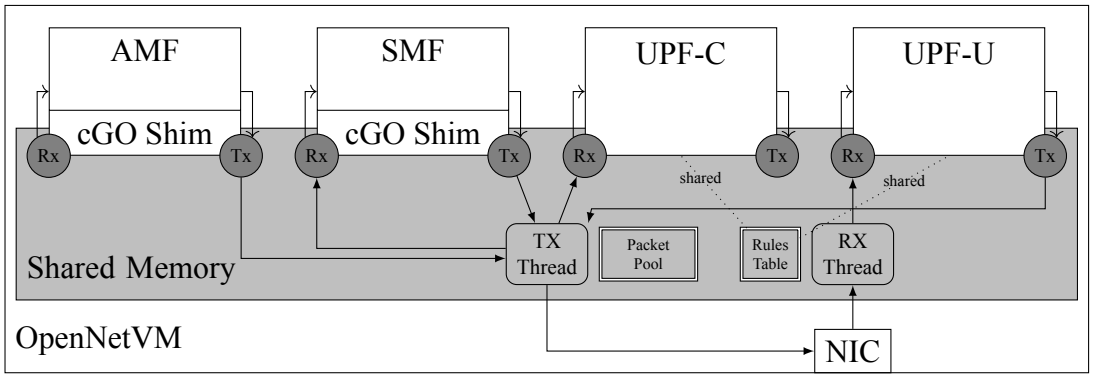


圖 3.10: LH5GC NF 架構

3.5 UPF 開發細節

在用户端的開發上有別於 free5GC 使用 gtp5g 這個 kernel module 開發，LH5GC 提供了一個使用者空間核心旁路的 UPF。雖然使用 gtp5g 可以重複使用核心提供的網路疊 (network stack)，且可以直接在核心空間進行用户封包的封裝或解封，避免重複複製到使用者空間，但還是會遇到章節 2.3所提到效能損耗，另外藉由核心開發的封包處理會需要透過中斷呼叫來溝通 [37] 與重複記憶體複製等效能瓶頸，而 UPF 又是核心網路用户端中唯一且最重要的 NF，因此我們認爲更應該對其作效能上的加速。

相對於高度核心依賴的 gtp5g-UPF，我們著重發展於軟體虛擬化的 UPF 功能，將 UPF 拆成兩個子部件：UPF-C 與 UPF-U，UPF-C 負責控制訊號的接收處理，而 UPF-U 則負責對用户封包進行規則查找與轉發，這兩個子部件分別會以不同的 OpenNetVM NF 包覆。如圖 3.10，當 SMF 向 UPF 發起控制訊號時，會透過 PFCP 協定傳送至 UPF-C，由於 LH5GC 是透過共享記憶體作相同節點跨 NF 的溝通，我們決定沿用的 PFCP 的封裝格式但捨棄其底層 UDP 的傳輸協議，直接將封包放置於記憶體上提供互相存取以提升溝通效率。當 UPF-C 收到 PFCP 訊息後，會根據 TLV 格式將 PFCP 解碼，並小心

的將內部資訊翻譯後放置於 UPF-C 與 UPF-U 可共同存取的地方。如圖 3.10 與圖 3.11 所示，UPF 會將上行 (UL) 與下行 (DL) 的規則分開儲存，各自維護一個 User Session 表格，在 User Session 表格內會存有各自的 Session Context，而這些上下文又會透過 TSS 資料結構儲存各種內的規則用以接下來的規則查詢。

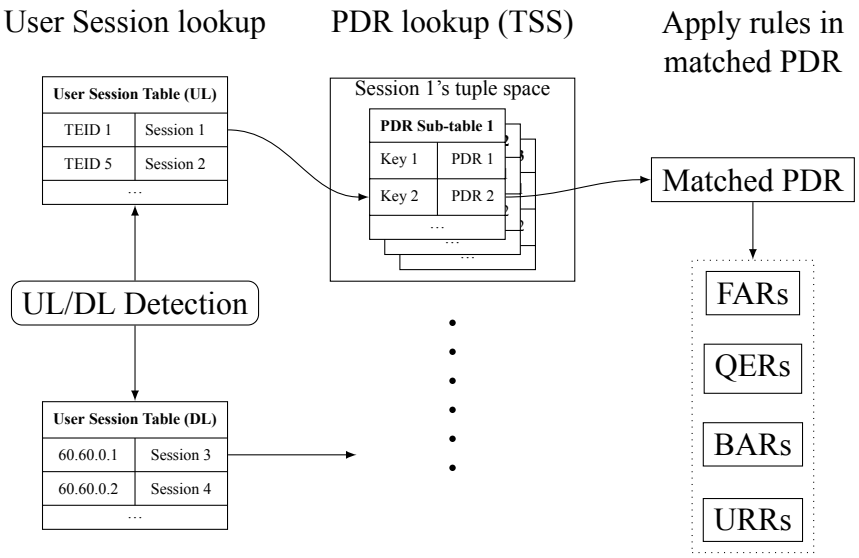


圖 3.11: LH5GC UPF 流程

當用户端封包進入到 LH5GC 系統時，OpenNetVM 的 NF Manager 會協助將封包送進 UPF-U 進行用户端封包處理。由於 UPF-C 與 UPF-U 透過共享記憶體的方式共用 Session Context，於 SMF 送至 UPF-C 的資訊 UPF-U 可直接查詢，因此當封包進入時，UPF-U 無須另外與 UPF-C 溝通便可進行以下動作查找相應的規則並予以應用 (見圖 3.11)：

1. 上下行流向偵測：UPF-U 會透過封包的目標 IP 決定其網路流向是屬於上行 (UL) 或下行 (DL) 的封包。
2. User Session 查找：對於上行封包，UPF-U 會透過 TEID 查詢相對應的 User Session，反之對於下行封包，UPF-U 會利用目標 IP 屬於那一個 UE 的決定其所屬 User Session。
3. PDR 搜尋：由於 User Session 內會存有一個或多個 PDR 規則，UPF-U 會透過 TSS 算法快速搜尋出封包所屬的 PDR，而當找到對應 PDR 後，便可連結出 PDR 所連結的其他規則 (例如 FAR、QER、BAR、URR)。
4. 執行規則：在收集到所有規則狀態後，UPF-U 會負責以符合這些規則的方式處理此封包。

另外爲了 Paging 或 Handover 等流程需要，UPF-U 內部也有實作封包緩衝 (buffer) 的機制。我們設計將緩衝空間根據 UE Session 個別存放於 UPF-U 的記憶體空間內，當封包從 NF 的 receive ring 送入 UPF-U 時，若查找出的規則內有 buffer 這個應用規則 (apply action)，UPF-U 會將封包存入所屬 UE Session 的緩衝內，反之若是其它規則，則 UPF 會將規則包裝進 packet descriptor 的元資料內並送入 transmit ring 中處理。

而當 SMF 更新規則把 buffer 改成其他諸如 forward 或 drop 的規則後，UPF 會再度把封包從緩衝取出並包入相對應規則的 packet descriptor，而後送入 transmit ring 之中處理。另外值得注意的是，爲了保證送出的封包順序正確，在儲存進緩衝時需要按照順序儲存。

四、 實驗與效能評估

爲驗證本論文設計之方法，將 free5GC 核心網路移植至 OpenNetVM，使用其所提供之 API 能有效提升核心網路之效能，本實驗分別對原生之 free5GC 核心網路部屬於 Ubuntu Linux，與經過 OpenNetVM 移植過後之 free5GC 核心網路一樣部屬於 Ubuntu Linux，分別對其控制端 (control plane) 與用户端 (user plane) 之效能進行測試，最終比對其結果，以驗證經過 OpenNetVM 移植過後的 free5GC 核心網路確實能有效提供更好的性能。

Baseline: 實驗過程我們會分別對原生之 free5GC 與經過移植至 OpenNetVM 的 LH5GC 進行註冊流程、連線建立流程、與換手流程等三個流程作測試，並比較其控制層的效能分析，另外我們也會建立連線後的用户端進行頻寬測試，最後我們也會比較基於 Berkeley Socket 與基於 OpenNetVM shared memory 的 SBI 的效能差別。

4.1 實驗環境設定

我們所使用的硬體與軟體規格如下表 4.1。

節點	項目	規格、型號、說明
Node 1 Traffic	CPU	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
	Core No.	4
	NIC	Intel Corporation Ethernet Controller X710/X557-AT 10GBASE-T
	OS	Ubuntu 18.04.5 LTS
	Kernel	5.4.0-60-generic
	Software	MoonGen Packet Generator [38]
Node 2 CN	CPU	Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz
	Core No.	8
	NIC	Intel Corporation Ethernet Controller X710/X557-AT 10GBASE-T
	OS	Ubuntu 20.04.1 LTS
	Kernel	5.4.0-65-generic
	Software	free5GC v3.0.4 || LH5GC RAN UE Simulator
Node 3 DN	CPU	Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz
	Core No.	4
	NIC	Intel Corporation Ethernet Controller 10G X550T
	OS	Ubuntu 18.04.5 LTS
	Kernel	5.4.0-62-generic

表 4.1: 系統環境參數

我們的實驗在邏輯上 (logically) 將系統分爲三個部件，邊緣端 (edge)、核心網路 (core network)、與數據網路 (data network)。邊緣端在眞實網路下即是手機、基地臺與相關部件，以下會以 UE-RAN simulator 代稱。核心網路如章節 refchapter:background 所講解的，是第五代行動通訊網路中，放置於機房 (data center) 中，負責統籌處理控制端 (control plane) 資訊與轉發 (forward)、封裝 (encapsulate)、解封裝 (decapsulate) 使用者端 (user plane) 資訊。而數據網路 (data network)，則是泛指傳統網路 (Internet)，即爲大多數時候，手機使用者想要存取的服務之所在位置。

爲了準確的區分流量產生與流量轉送的中央處理器 (以下簡稱 CPU) 使用率，避免單一 CPU 需要同時產生流量與處理流量，我們的測試方法將流量產生器 (traffic generator)、核心網路、流量接收者 (traffic receiver) 分別部屬於不同的機器上，讓三者得以完全妥善使用三顆不同之 CPU。

因此在務實面 (physically) 上將系統切分成三個節點，分別模擬使用者端流量發送、基地臺與核網處理、資料網路回覆 (圖 4.1)，在節點與節點中我們都是使用 Intel 的 10 Gbps 雙孔網卡與 CAT-6A RJ45 網路線連接。在 Node 1 上面我們使用 MoonGen [39] 軟體來產生流量。在 Node 2 上面我們分別部屬原生的 free5GC v3.0.4 與修改過後的 LH5GC 作爲核心網路測試，另外比較特別的是，我們也把自行開發的 RAN UE 模擬器部屬在 Node 2 上面而非 Node 1 上，此模擬器會模擬 5G SA 的控制訊號與核心網路建立連線。而在 Node 3 上主要模擬資料網路回復封包，但若是測試單純下行流量，則會在 Node 3 上啓動 MoonGen 來產生流量。

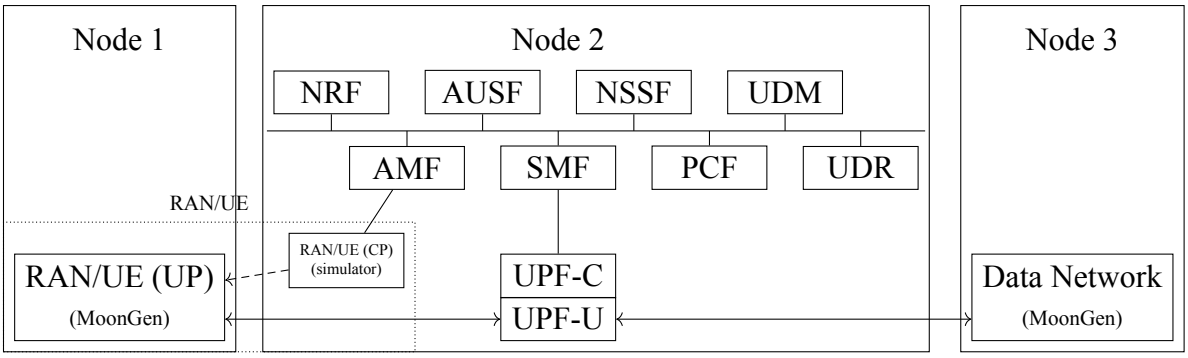


圖 4.1: 實驗環境節點

4.2 用户層效能分析

4.2.1 上行流量分析

在用户端的測試中，我們首先做了上行流量測試，分別使用不同大小的 ping 封包從 Node 1 藉由 Node 2 送至 Node 3，在這個測試中關閉 Node 3 ping 的回復功能 (response)，觀察 Node 3 接收到的流量。

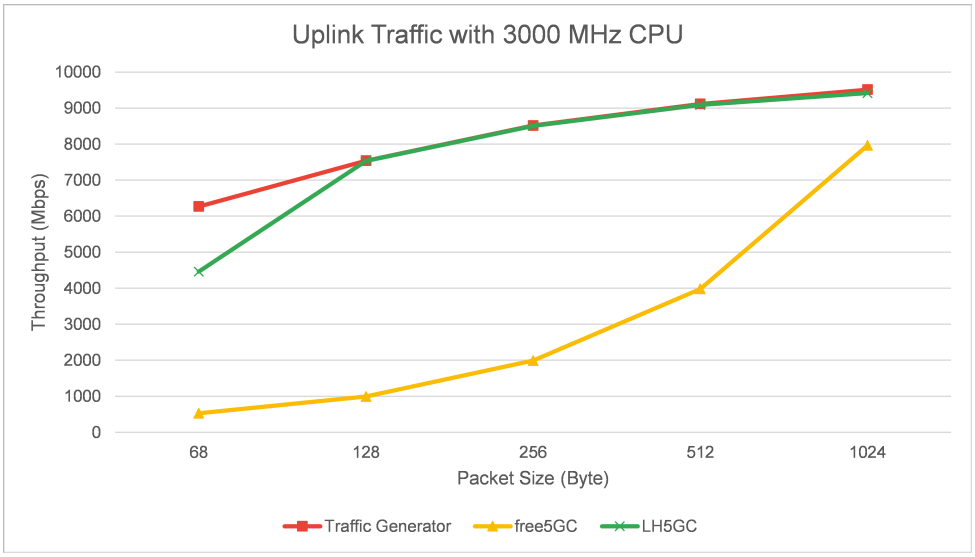


圖 4.2: 用户層上行效能比較

圖 4.2的縱軸是吞吐量，單位爲每秒處理的總位元組，固定 CPU 時脈於 3000 MHz 並比較不同封包大小的影響。由圖中可以很明顯的看出 LH5GC 用户端流量處理能力遠高於 free5GC。在封包大小爲 68 位元組 (Byte) 時，LH5GC 的吞吐量會是 free5GC 的十一倍之多，另外在封包大小大於 128 位元組後，LH5GC 的吞吐量會幾乎逼近原始產生的流量，足見得其效能並沒有達到 LH5GC 的流量上限，而是受制於網卡的處理能力。另外隨著封包大小的提升，free5GC 的吞吐量會有明顯的提升，其歸因於核心 (kernel) 實作的單位時間封包處理數已達到其固定量。

4.2.2 來回流量比較

在這個測試中，我們比較 LH5GC 與 free5GC 上下行流量，除了關閉 Node 3 ping 回復功能的測項外，也測試了開啓回復功能後，Node 1 接收到的流量。

圖 4.3在固定 CPU 時脈於 3000 MHz 並使用 68 位元組的封包，比較 LH5GC 與

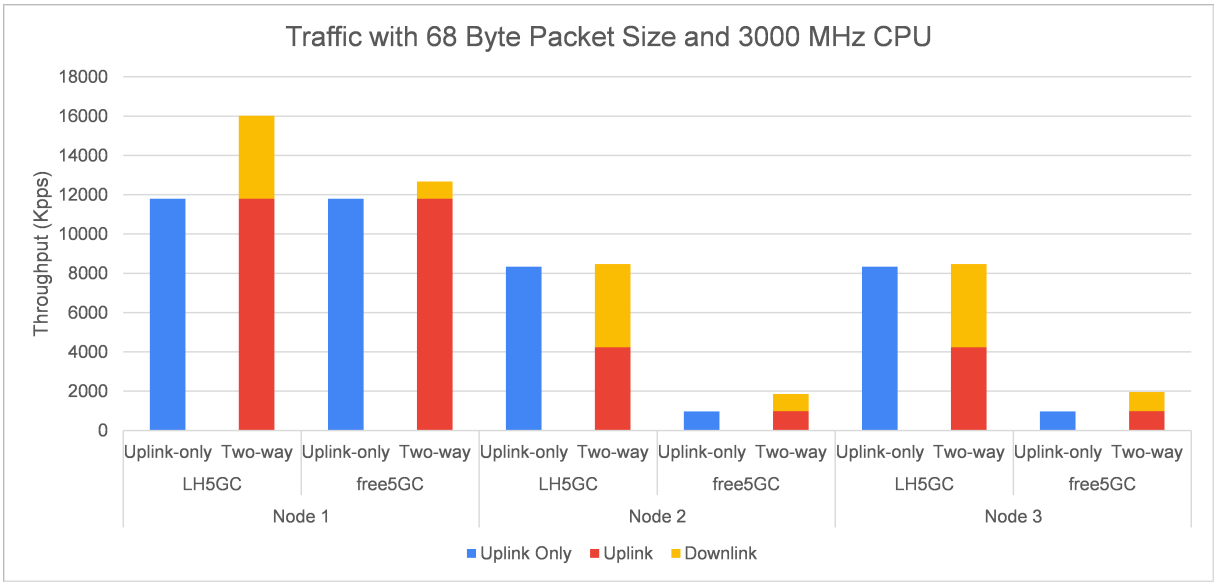


圖 4.3: 用户層上下行比較

free5GC 分別在 Node 1、Node 2、與 Node 3 的上下行流量，藍色長條顯示當只有測試上行時的流量，而紅、黃色長條則是在同時有上下行流量時，其各自的流量大小，縱軸爲吞吐量，這裡單位取每秒處理的封包數量。可以看到當流量從 Node 1 被產生到送到 Node 2 處理時，相較於 free5GC，LH5GC 所能承受的封包數量足足八倍有餘，而當封包回復至 Node 1 時，LH5GC 的流量會是原本送出時的 $\frac{1}{3}$，而 free5GC 則會是降低成原來的 $\frac{1}{13}$ (比較 Node 1 紅、黃直條的差距)。

另一個有趣的觀察是在 Node 2 上比較單獨上行與上下行流量分配的差異，LH5GC 在單獨上行可處理的封包量約等於同時處理上下行的封包量，而 free5GC 則是可以處理加倍的吞吐量。推測是由於受益於 Kernel Socket 的實作，在同時處理上下行時可以分別使用不同的 Kernel Thread 處理，而 LH5GC 由於是透過 DPDK 實作，會搶佔單獨一個 CPU 核心進行網卡封包的輪詢，因此若需要同時處理上下行，亦會受限於單一核心的輪詢能力。

4.2.3 CPU 時脈影響

由於在測試的過程中，我們發現 CPU 時脈會對流量產生可觀的影響，因此我們也嘗試測試 CPU 時脈與流量的關係。

圖 4.4透過 Linux 系統指令限制 CPU 的時脈，可以看出隨著時脈增加，LH5GC 單位時間處理的封包數明顯增長，而對 free5GC 的影響則較不明顯。透過這張圖可以發現，

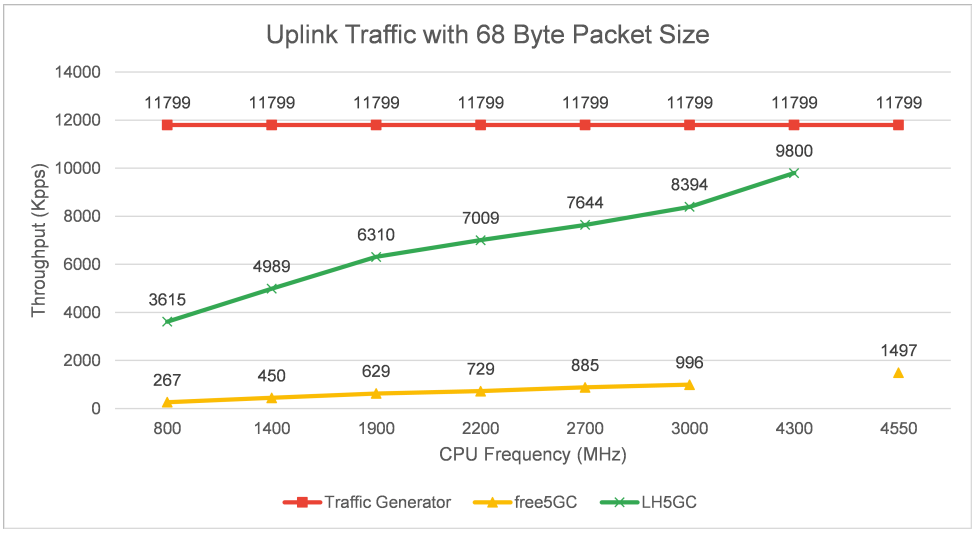


圖 4.4: 用户層受時脈影響分析

如果有足夠的 CPU 時脈，LH5GC 將可不斷增加吞吐量。

4.3 控制層效能分析

4.3.1 PFCP 延遲差異

我們測試當 UE 建立連線以及換手時所會涉及到的 PFCP 訊息延遲，包含建立連線上行路徑的 Session Establishment、建立下行路徑的 Session Modification (1)、換手時改變上行路徑時的 Session Modification (2)、以及換手時改變下行路徑的 Modification (3)。由於 LH5GC 基於 OpenNetVM 平臺的 CPU 配置，只會使用到 3 個 CPU 核心，因此在 free5GC 的測試時我們也跑了兩種測試，其一是在同一系統上開啓 3 個 OpenNetVM 平臺上的無功能 NF，模擬 LH5GC 的核心使用狀況，另一個則是直接透過系統指令鎖住核心數，讓 free5GC 只能在三個核心上運行。

圖 4.5的縱軸是時間，取毫秒爲單位。可以看出 free5GC 在各信息的平均延遲皆大於 LH5GC 的測試，在 Establishment 訊息上 free5GC 多 LH5GC 1.8 倍的延遲時間、在 Modification 1 上多了 1.5 倍、在 Modification 2 上多了 2.5 被、而在 Modification 3 上多了 1.9 倍的延遲。另外具有 3 個空 NF 的延遲會是最大的，我們認爲是因爲雖然有 3 個 NF 並沒有功能，但依然會消耗到 CPU 的運算資源。

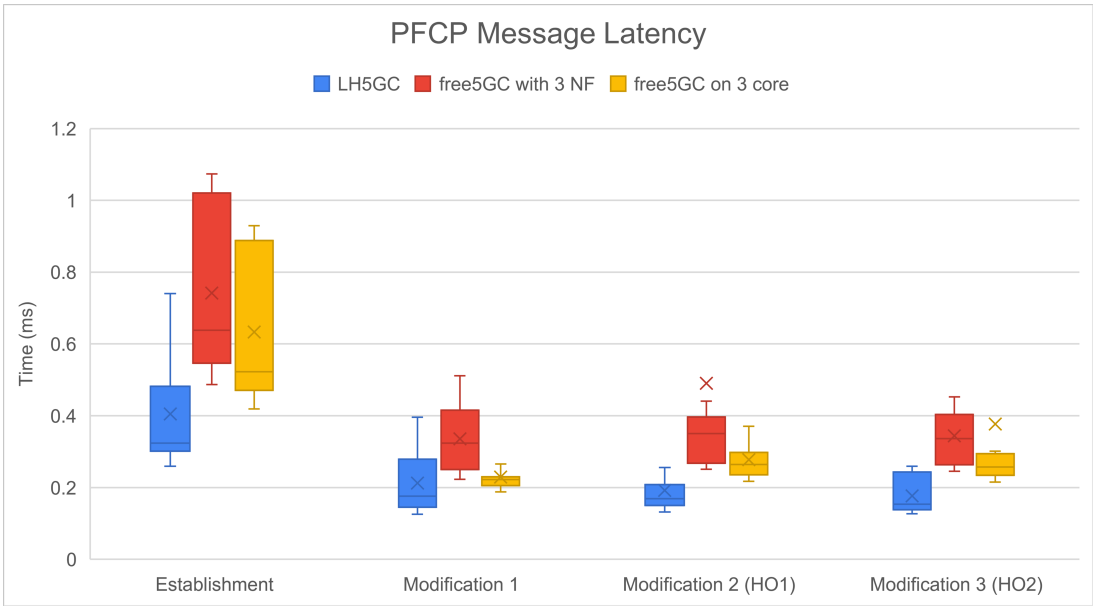


圖 4.5: PFCP 訊息延遲比較

4.3.2 SBI 延遲差異

在 SBI 延遲測試中，我們選擇測試 SMContextCreate 此單條訊息，此訊息是由 AMF 送至 SMF，用以通知 SMF 開始建立連線的控制訊息。

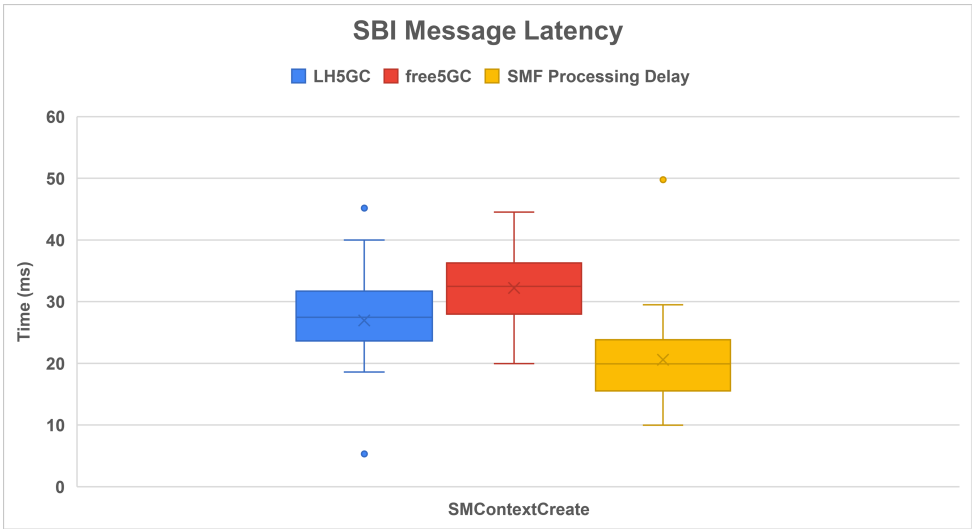


圖 4.6: SBI 訊息延遲比較

圖 4.6可看出 LH5GC 透過共享記憶體的方式溝通的效能略優於 free5GC 透過傳統 Kernel Socket 的傳輸，但似乎不太明顯。但若我們嘗試將 NF 的 Processing Delay 扣除 (見表 4.2)，則 LH5GC 在 SMContextCreate 訊息的傳播延遲 (propagation delay) 平均會是 6.38 ms，而 free5GC 的平均則是 11.65，可見除了減少溝通延遲，NF 本身的運算速度

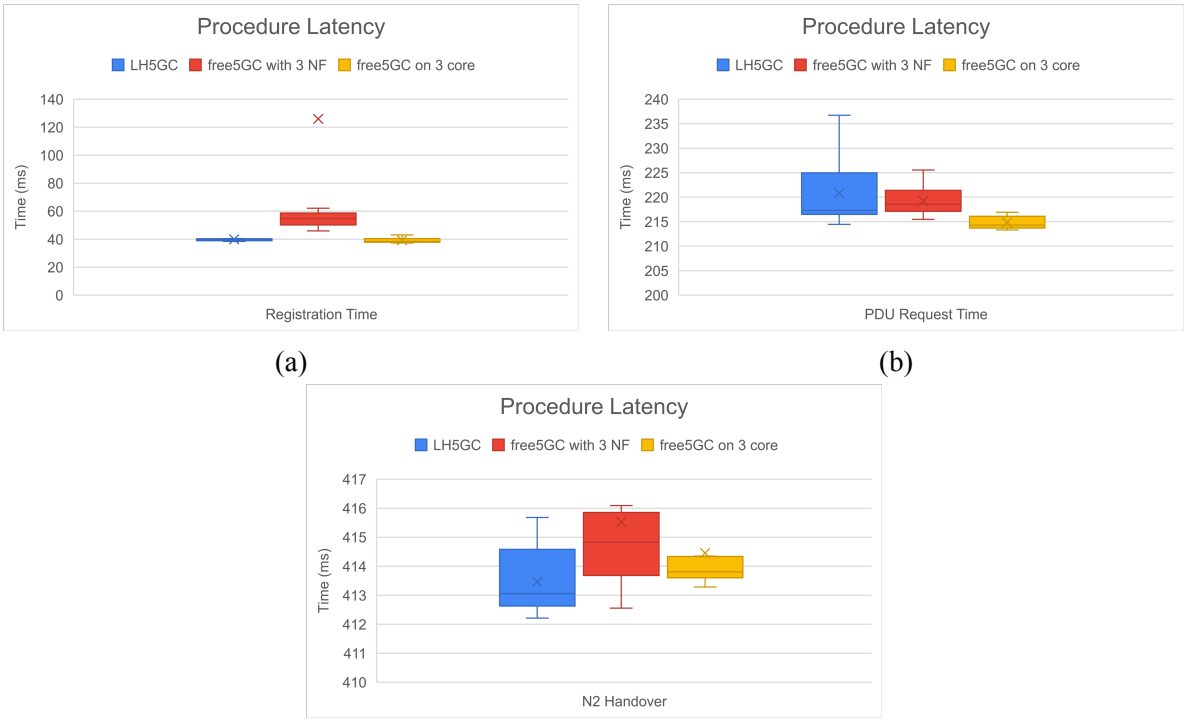
	LH5GC	free5GC	Processing Delay
Average Delay (ms)	26.95673585	32.2257988	20.57215
SMF Propagation Delay (ms)	6.38458545	11.6536484	

表 4.2: SBI 傳播延遲

亦會對總體延遲產生影響。

4.3.3 控制流程延遲比較

最後，我們比較控制端三個完整流程的延遲：註冊流程、連線建立流程、以及換手流程。同樣的我們比較的是 LH5GC 、free5GC 與在同一系統上開啓 3 個 OpenNetVM 平臺上的無功能 NF、以及 free5GC 只能在三個核心上執行的版本。



(a) (b)

(c)

圖 4.7: 控制流程延遲差異

由圖 4.7可以看出就整體控制端流程，LH5GC 的平均值會稍微比 free5GC 的兩個測試版本好，然而差別並不大，就其原因可以從觀察 SBI 單一訊息的延遲觀察中發現 (表 4.2)，雖然 LH5GC 在訊息傳遞的傳播延遲上好過 free5GC，但就處理延遲 (processing delay) 與傳播延遲 (propagation delay) 的比例來看 ($26:6 \approx 4:1$)，並沒有獲得很大比例的效益。另外，控制訊號的傳遞延遲需要透過更多訊息累積才能獲得比較

好的效益，而目前 LH5GC 僅有在 PFCP 與 SMContextCreate 介面上完成共享記憶體的使用，若可將所有 SBI 移植至共享記憶體上，將會帶來更大的效益。

五、 結論

本篇論文提出並實作 LH5GC 這個建立在網路功能虛擬化下的低延遲高流量 5G 核心網路平臺，透過 OpenNetVM 平臺的移植與對同一節點部屬的觀察，在不改變 3GPP 定義的 5G 架構與溝通流程下，提供了包含控制端與用户端的效能都比以往 3GPP 框架下基於 Berkeley Socket 的 5G 核心網路高的 5G 核心網路框架。在控制端，我們利用共享記憶體的加速，實踐於 SBI 與 N4 介面上，獲得了比傳統架構低一半的傳遞延遲。在用户端上，透過利用 DPDK 越過核心網路疊的效能損耗以及 TSS 的規則加速查找，我們可以比傳統架構多 11 倍的吞吐量。最後，LH5GC 提出了更有效率的換手流程，利用 UPF 緩衝取代 3GPP 架構中提供的封包間接轉發機制，避免了過長的封包轉發路徑。

六、 未來展望

藉由本篇論文的實作與實驗可知論文的架構具備可行性，但尚有許多細節需要被改善。該如何提供此框架更好的擴展性值得我們討論，LH5GC 基於 OpenNetVM 平臺開發，然而目前控制訊息僅能使用共享記憶體而用户訊息則只受惠於 DPDK，是否可以使平臺自動選擇最佳化的通訊頻道？另外像是 SBI 的最佳化，目前只完成在 SMContextCreate 上的概念驗證 (PoC)，若可以完整部屬於全部 SBI 介面上，相信有助於大幅降低控制端的延遲。而又透過傳播延遲與執行延遲的比較 (表 4.2)，我們知道除了通訊管道的加速外，NF 本身的運算速度亦是另一個可以大幅提升效能的地方，而像是 CGO 的使用，雖然方便快速開發，但是其效能損害卻遠比想像中的大 [40]，爲加速執行效率，修改 cgo 或直接改用組語階層連接將可以是一個發展方向。於章節 4.2.2中有提到，目前 LH5GC 的用户端受限於 DPDK 的單核 CPU 輪詢，未來勢必可以考慮朝多核心的 CPU 輪詢研究，以達更佳的 CPU 使用效率。

參 考 文 獻

[1] ITU-R, “IMT Vision —Framework and overall objectives of the future development of IMT for 2020 and beyond,” 09 2016. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf

[2] 3GPP, “System architecture for the 5G System (5GS),” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 09 2018, version 15.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[3] 3GPP, “General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.401, 12 2017, version 8.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=849

[4] OpenAPI Generator. [Online]. Available: https://github.com/OpenAPITools/openapi-generator

[5] 3GPP, “Interface between the Control Plane and the User Plane nodes,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.244, 09 2018, version 15.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3111

[6] 3GPP, “Procedures for the 5G System (5GS),” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.502, 09 2018, version 15.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145

[7] NextEPC. [Online]. Available: https://nextepc.org/

[8] Open5GS. [Online]. Available: https://open5gs.org/

[9] OPENAIR-CN. [Online]. Available: https://github.com/OPENAIRINTERFACE/openair-epc-fed

[10] 3GPP Release 15 (R15). [Online]. Available: https://www.3gpp.org/release-15

[11] OpenAirInterface. [Online]. Available: https://openairinterface.org/

[12] OPENAIR-CN-5G. [Online]. Available: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed

[13] OpenAirInterface 5G Core. [Online]. Available: https://openairinterface.org/oai-5g-core-network-project/

[14] Internship-5GCN. [Online]. Available: https://github.com/bubblecounter/Internship-5GCN

[15] Metaswitch. [Online]. Available: https://www.metaswitch.com/products/fusion-core

[16] free5GC. [Online]. Available: https://www.free5gc.org/

[17] OpenUPF. [Online]. Available: https://github.com/5GOpenUPF/openupf

[18] upf-p4-poc. [Online]. Available: https://github.com/801room/upf_p4_poc

[19] User Plane Gateway (UPG) based on VPP. [Online]. Available: https://github.com/travelping/upg-vpp

[20] 3GPP, "Architecture enhancements for control and user plane separation of EPC nodes," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.214, 06 2018, version 15.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3077

[21] FD.io Vector Packet Processor. [Online]. Available: https://fd.io/

[22] upf-xdp. [Online]. Available: https://github.com/801room/upf-xdp

[23] Awesome 5G. [Online]. Available: https://github.com/calee0219/awesome-5g

[24] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward Low-Latency and Ultra-Reliable Virtual Reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 04 2018.

[25] “Cloud AR/VR Whitepaper,” 10 2019. [Online]. Available: https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/

[26] (2019, 07) Bandwidth required for HD FHD 4K video streaming. [Online]. Available: https://www.synopi.com/bandwidth-required-for-hd-fhd-4k-video/

[27] R. Ford, M. Zhang, M. Mezzavilla, S. Dutta, S. Rangan, and M. Zorzi, “Achieving Ultra-Low Latency in 5G Millimeter Wave Cellular Networks,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 196–203, 03 2017.

[28] Y. Li, Z. Yuan, and C. Peng, “A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks,” in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’17. New York, NY, USA: Association for Computing Machinery, 10 2017, pp. 56–69. [Online]. Available: https://doi.org/10.1145/3117811.3117838

[29] U. Cisco, “Cisco Annual Internet Report (2018—2023) White Paper,” 03 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[30] 3GPP, “5G System; Technical Realization of Service Based Architecture; Stage 3,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.500, 03 2019, version 15.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3338

[31] gtp5g. [Online]. Available: https://github.com/free5gc/gtp5g

[32] openNetVM. [Online]. Available: http://sdnfv.github.io/onvm/

[33] J. Hwang, K. K. Ramakrishnan, and T. Wood, “NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms,” *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, 03 2015.

[34] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood, “OpenNetVM: A platform for high performance network service

chains," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, ser. HotMIddlebox '16. New York, NY, USA: Association for Computing Machinery, 08 2016, pp. 26–31. [Online]. Available: https://doi.org/10.1145/2940147.2940155

[35] V. Srinivasan, S. Suri, and G. Varghese, "Packet Classification Using Tuple Space Search," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '99. New York, NY, USA: Association for Computing Machinery, 08 1999, pp. 135–146. [Online]. Available: https://doi.org/10.1145/316188.316216

[36] Chia-An Lee, Hao-Tse Chu, Hung-Cheng Chang, and Vivek Jain, "onvmNet," 2021. [Online]. Available: https://github.com/nctu-ucr/onvmNet

[37] J. C. Mogul and K. K. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-Driven Kernel," *ACM Trans. Comput. Syst.*, vol. 15, no. 3, p. 217—252, 08 1997. [Online]. Available: https://doi.org/10.1145/263326.263335

[38] P. Emmerich, "MoonGen Packet Generator," 2020. [Online]. Available: https://github.com/emmericp/MoonGen

[39] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, 10 2015, pp. 275–287. [Online]. Available: https://doi.org/10.1145/2815675.2815692

[40] D. Cheney. (2016, 01) cgo is not Go. [Online]. Available: https://dave.cheney.net/2016/01/18/cgo-is-not-go