電腦網路概論

Introduction to Computer Network

網路？

訊息的傳遞

- 假設現在有兩個人 Alice 跟 Bob，Alice 想要找 Bob 一起玩遊戲，Alice 住在北邊的洞穴、Bob 住在南邊的洞穴，
  他們可以怎麼傳遞訊息？

訊息的傳遞

- Alice 自己走過去跟 Bob 說要找他玩
  - Alice 自己辛苦地走過去跟 Bob 說要找他玩

    如果他們每次要說話都要走這麼遠，那也太累了！
- Alice 找他朋友 rrruuu56 去跟 Bob 說 Alice 要找他玩
  - 這樣 Alice 不用自己走，但還是要等朋友傳話，而且朋友也得走

訊息的傳遞

- 這兩種方法還受到一個限制：距離
  - 如果兩個人都住在很遠的地方，那麼傳遞的效率就非常低
- 因此，現代人就找到了一個解決方法：網路 (Network)

網路的構成

網路的構成

- 設備 (Devices)
  - 終端設備：手機、平板、伺服器
  - 中介設備：路由器 (Routers)、交換機 (Switchs)
- 傳輸媒介
  - 有線: 網路線、光纖
  - 無限：WiFi、藍芽

網路的構成

- 今天 Alice 跟 Bob 想要一起玩，他們該如何傳達一個「對方看得懂的訊息」？
  - 傳輸一種對方看得懂的語言 (例如：中文、英文)
- 今天如果是網路：
  - 想想看，世界上這麼多設備，
    你在傳輸網路資料時要怎麼確保全世界的設備都看得懂？

協定 (Protocols)

- 解決問題的方式：制定一種全世界遵守的網路通訊規則！
- 這一個規則我們就稱為：協定 (Protocols)
- 協定的種類有非常多：
  - HTTPS, HTTP, TCP, UDP, ICMP 等

網路的架構

網路的架構

- 區域網路 (LAN)
- 廣域網路 (WAN)
- 有時候我們會簡單稱呼為「內網」與「外網」
- 可以想像網路世界是好幾個社區組成的
  - 每一個社區就是一個「區域網路」
  - 社區外就是「廣域網路」

你是如何上網的？

網路的大門：數據機

- 大家常聽到的「小烏龜」
- 負責接收來自電信公司提供的網路訊號
  - 轉變訊號，提供給電腦
  - 轉變訊號，透過網路線傳輸

路由器 (Router)

- 從數據機接收網路，並分發網路給其他設備
- 數據機也可以直接給予設備網路，但有時候我們的設備很多，我們就會透過路由器，讓路由器可以分發更多網路給其他設備

IP 位址

IP (網際網路協定)

- 在網路世界中，IP 是一種地址，用來識別設備
- 只要能連上網路的設備都會被分發到一個屬於自己的 IP 位置

IPv4 (Internet Protocol Version 4)

- 由四個數字組成 (Ex. 192.168.1.1)
- 這四個數字的範圍介於 0 ~ 255 之間
- 受限於數字範圍，現在 IPv4 已經快要用盡了！

公有 IP 與 私有 IP

- 公有 IP: 在網路世界中「獨一無二」的 IP 位置
- 私有 IP: 在區域網路中 (內網) 所使用的 IP
- 這樣就不需要每一個設備都需要一個公有 IP，可以節省 IP 的使用

公有 IP 與 私有 IP

- 綜合我們剛剛學到的：
  - 意味著在數據機以內，設備之間溝通都使用私有 IP
  - 出了數據機以後就是使用公有 IP

私有 IP

- 私有 IP 可以利用前 1 個或 2 個數字來分辨：
  - 10.0.0.0 ~ 10.255.255.255
  - 172.16.0.0 ~ 172.16.255.255
  - 192.168.0.0 ~ 192.168.255.255

Port

- IP 的服務窗口
- 一個設備可能有很多種服務：
  - 網頁服務 (HTTP, HTTPS)
  - 電子郵件 (SMTP)
  - 下載 (FTP)
- 為了區分這三種服務，每一個都會需要一個 IP 位置
- 這樣太浪費了！因此就有了 Port 的概念

Port

- 透過 Port 將每一種服務切分到一個特定窗口 (數字)
- 假設目前的伺服器 IP 位置：140.116.133.22
- 所有服務：
  - 網頁服務 (Port: 2222)
  - 下載服務 (Port: 3333)
  - 電子郵件 (Port: 4444)
- 我要指定使用網頁服務就只要將 IP 加上 :[Port 號碼]
  - 140.116.133.22:2222

Port

- Port 的數字介於 (0 ~ 65535)
- 某些特定服務有預設 Port 位置
  - ssh (Port 22)
  - http (Port 80)
  - https (Port 443)

NAT 轉譯

NAT 轉譯

- 我們先目前的網路架構：
  - 數據機 1 台
  - 路由器 1 台
  - 電腦 3 台
- 這三台電腦會有同樣的公有 IP (Public IP)
- 但會有三個不同的私有 IP (路由器分配給每一個電腦一人一個)

NAT 轉譯

- 公有 IP: 140.116.80.22
- 電腦 1: 192.168.0.2
- 電腦 2: 192.168.0.3
- 電腦 3: 192.168.0.4
- 還記得 192.168 開頭的 IP 是什麼 IP　ㄇ ？

NAT 轉譯

- 架構：數據機 -> 路由器 -> 三台電腦
  - 因此每一台電腦對外連線的順序：
    - 電腦 -> 路由器 -> 數據機
    - 公有 IP 都是同一個！

NAT 轉譯

- 路由器的重要功能：NAT 轉譯
- 在資料出內網之前，路由器會負責將私有 IP 轉成公有 IP
- 這一個轉換的過程稱為：NAT 轉譯

NAT 轉譯

- 假設目前電腦 1 (192.168.0.2) 對外傳輸資料
- 經過路由器後，路由器會將 IP 換成 140.116.80.22 才發送出去

NAT 轉譯

- 假設剛剛發送資料後，對方有回應資料回來，NAT 也會發生！
- 由於對方會知道是 140.116.80.22 發送的資料，因此回應資料也會回傳到這個 IP 位置
  - 這個時候路由器要正確的知道這一筆資料是要給電腦 1 的也就是 192.168.0.2
  - 所以路由器又會需要將 140.116.80.22 的 IP 換回 192.168.0.2

NAT 轉譯

- 路由器會記錄一個轉譯表
- 這樣對方回應時才能知道要將此回應交給誰

NAT 轉譯

- 不知道大家小時候有沒有遇過架 Minecraft 的難題？
- 會有這樣子的難題的元凶其實就是 NAT！

NAT 轉譯

- 原本的情況：
  - 我們發送資料並且將傳輸資訊記錄在轉譯表
  - 對方回應時透過轉譯表確認該轉交給誰
- Minecraft 的情況：
  - 對方直接傳輸資料到路由器
  - 路由器不知道要將這筆資料給誰

封包

封包

- 用來統稱在網路中傳輸的資料，可以想像是一種包裹
- 假設今天你要傳輸一個文件，這一個文件可能很大：
  - 將文件分成好幾份，一份一份傳輸
  - 這每一份東西就是「封包」

封包的內容

- 封包就跟我們寫信一樣，信封上有基本的資訊：
  - 來源 IP (寄件人)
  - 目的 IP (收件人)
  - 來源 Port
  - 目的 Port
  - 序號 (這個等等會說)
  - 其他資訊 (Ex. checksum)

封包的序號

- 還記得我們會將一份大資料切成好幾個封包傳輸 ㄇ？
- 當對方收到這些封包後，自然會需要將這些封包重組
  - 為了協助對方以正確的順序重組資料，因此每一個封包上都會有一個特定的序號！

DNS

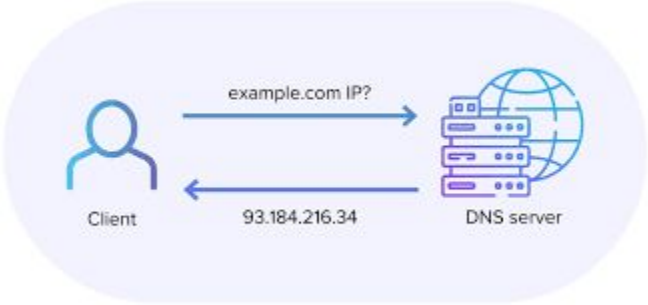
Domain Name System (DNS)

- 還記得每一個設備都會有一個自己的 IP 位置 ㄇ ？
- 如果某一個網頁服務架設在 140.116.246.40

  那麼我們在網址中輸入 140.116.246.40 就可以連上對應網站
- 但是，你有發現我們平時輸入的網址都是一串英文字組成的 ㄇ ？
  - Example: www.google.com

Domain Name System (DNS)

- 你的電腦怎麼知道這些文字是對應到什麼 IP？
  - 透過 DNS！
- DNS 會幫我們把對應的網址轉為對應的 IP 提供我們連線



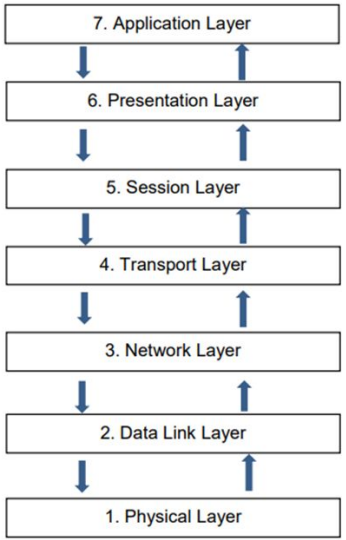
OSI 七層

OSI 七層

- 前面介紹了很多名詞，可以發現網路的傳輸需要經過非常多步驟
- 網路再進行傳輸資料時，其實是一個分工明確的產線！
  - 應用層 (Application Layer)
  - 表示層 (Presentation Layer)
  - 會話層 (Session Layer)
  - 傳輸層 (Transport Layer)
  - 網路層 (Network Layer)
  - 資料連結層 (Data Link Layer)
  - 實體層 (Physical Layer)

OSI 七層

- 發送資料時：從第 7 層開始包裝封包，不斷包裝直到第 1 層
- 接收資料時：從第 1 層開始拆開封包，不斷拆開封包到第 7 層



應用層

應用層

- 最貼近使用者的一層
  - 不干涉資料傳輸的方式
  - 提供「網路」給應用程式
  - 定義應用程式如何接入與使用網路，處理兩應用程式之間進行資料交換與協調

應用層的協定

- HTTP, HTTPS
- SMTP (電子郵件)
- FTP (檔案傳輸)

傳輸層

傳輸層

- 負責處理應用程式之間的資料傳輸
- 確保資料順利送達

傳輸層 - 分段與重組

- 將大資料分割，方便傳輸
- 將接收到的資料重組成原始資料

傳輸層 - Port

- 前面介紹的 Port 就屬於傳輸層的工作
- 傳輸層會透過 Port 來識別不同的應用程式與服務，並透過此 Port 的號碼將資料傳送到正確的目標

傳輸層 - 流量控制

- 防止接收方因突然收到大量資料而癱 瘓
- 傳輸層會透過一種協定 (Sliding Window Protocol) 來協調發送方與接收方之間的資料傳輸速度

傳輸層協定

- TCP & UDP
- 應該算是網路裡面最知名的兩個協定了！

封包遺失 (掉包)

- 網路的世界中難免發生封包遺失的問題，我們可稱為掉包
- 就類似現實生活郵差把包裹送不見了一樣
- 造成封包遺失的原因有非常非常多種，因此掉包是一種很常見也很正常的現象

封包遺失的影響 (例子)

- 玩遊戲卡頓、斷線
- 網頁讀取變慢
- 聽音樂斷斷續續

如何確保資料安全抵達？

- 既然封包難免遺失，那我們要如何保證封包有交到對方手上？
  - 可以想想看現實生活中寄信會如何防止遺失？

如何確保資料安全抵達？

- 既然封包難免遺失，那我們要如何保證封包有交到對方手上？
  - 可以想想看現實生活中寄信會如何防止遺失？
  - 限時掛號！
    - 追蹤包裹
    - 一定要對方簽收，不能只是丟信箱
- 在網路的世界中，也有類似的機制！

TCP

- TCP 協定會檢測是否發生掉包，並觸發重傳機制
  - 如果發現掉包，那麼就重寄一個，確保該資料送達對方手上

UDP

- 沒有重傳機制，掉包就算了，簡單暴力！
- 那為什麼我們會需要 UDP？

UDP

- 沒有重傳機制，掉包就算了，簡單暴力！
- 那為什麼我們會需要 UDP？
  - 速度快！
  - 對於某些即時性高的應用，發送一堆封包，
    偶爾掉幾個基本上不影響，且可以保持高傳輸效率
  - Example: 英雄聯盟 (LOL) 就是使用 UDP

TCP - 如何提供可靠的傳輸機制？

- 傳輸前建立連接 (就像是打電話給對方且對方接起來了)
- 傳輸後關閉連接 (掛電話)
- 可靠傳輸：
  - 確認機制
  - 重傳機制
  - 確保封包按照順序到達
  - 檢驗數據是否損壞

TCP - 確認機制

- 接收方確認接收無誤後要回傳一個「收到」訊息
  - 這個在網路的世界中我們稱為 ACK (Acknowledgement)
- 流程：
  - 發送方發送數據，每一個數據照順序發送並以序號區分
  - 接收方確認收到一段數據，並回報 ACK 封包給發送方

TCP - 確認機制

- 接收方確認接收無誤後要回傳一個「收到」訊息
  - 這個在網路的世界中我們稱為 ACK (Acknowledgement)
- 流程：
  - 發送方發送數據，每一個數據照順序發送
  - 接收方確認收到一段數據，並回報 ACK 封包給發送方
    - 這一個 ACK 封包會帶有一個 ACK Number，
      代表接收方下一個預期要收到的數據

TCP - 確認機制

- Example:
  - 接收方已收到 1000 ~1099 的數據,

    發回的 ACK Number 會是 1100
  - 表示接收方預期下一個數據要是 1100 開始

TCP - 確認機制

- 這邊要特別注意不要跟前面的「封包序號」 搞混！
  - 封包序號是針對整個封包
  - TCP 的序號是針對資料內容區分出來的號碼

電腦世界儲存資料的方式

- 電腦世界中以二進位來儲存資料，每一個位置我們稱為「位元」
- 一段完整個資料是由好幾個位元所組成
- TCP 的序號指的其實就是資料的位元

TCP - 確認機制

- Example:
  - 接收方已收到 1000 ~1099 的數據,

    發回的 ACK Number 會是 1100
  - 表示接收方預期下一個數據要是 1100 開始
  - 所以這裡的意思其實就是，已經收到位元 1000~1099 的數據,

    下一個收到的數據要從第 1100 位元開始

TCP - 重傳機制

- 我們已經知道確認機制了，但是發生掉包要怎麼重傳？
- 我們先來想想如何知道發生掉包？

TCP - 重傳機制

- 我們已經知道確認機制了，但是發生掉包要怎麼重傳？
- 我們先來想想如何知道發生掉包？
  - 你寄了一封信，過了一個月你朋友還沒收到
  - 關鍵因素：時間！
  - 當發現過了很久對方都沒跟你說已收到就表示可能發生掉包

TCP - 重傳機制

- TCP 傳輸每一段資料時都會 啟動計時
- 一定時間內沒有收到 ACK 就會認定這段資料寄丟了，觸發重傳
- 有沒有一種可能是：原本的封包沒有不見，只是單純傳太慢？

TCP - 重傳機制

- TCP 傳輸每一段資料時都會 啟動計時
- 一定時間內沒有收到 ACK 就會認定這段資料寄丟了，觸發重傳
- 有沒有一種可能是：原本的封包沒有不見，只是單純傳太慢？
  - 當然！TCP 一樣會觸發重傳機制，也就是說接收者會因此收到 2 段一樣的資料，TCP 本身有辦法處理並捨去掉重複的部分

TCP - 重傳機制

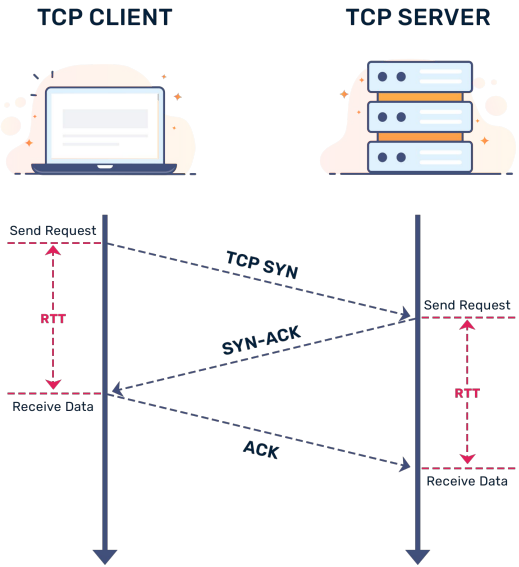
- 假設超過 x 單位時間會觸發重傳機制
- 那麼 x 應該要是多少才合適？

TCP - 重傳機制

- 在不同的網路環境下，網路的傳輸速率都不同
- x 會是一個需視當前網路情況調整的變數
- TCP 如何決定 x？

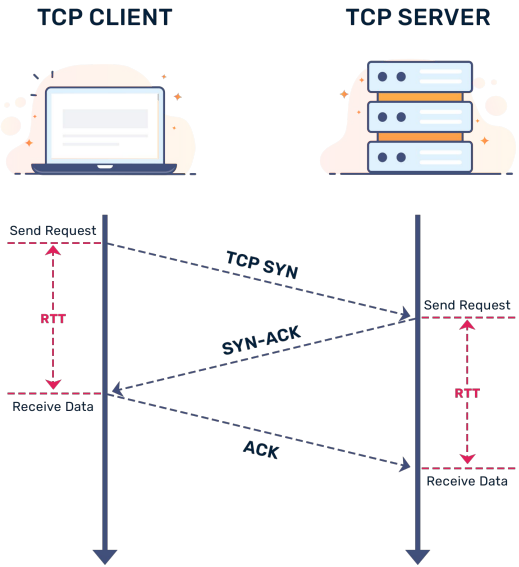
TCP - 重傳機制

- 這邊先介紹一個名詞：RTT (Round-Trip Time)



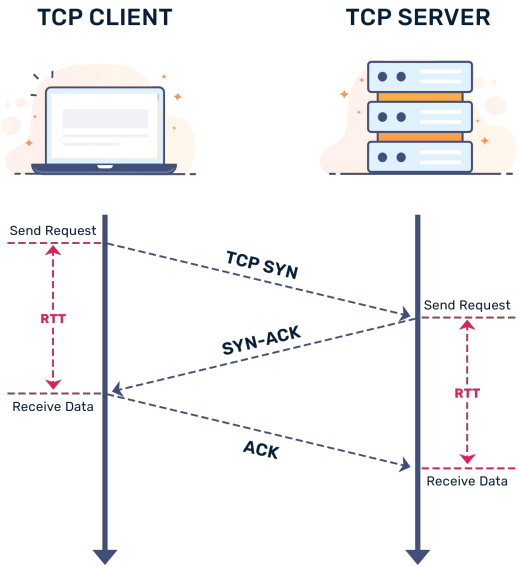
Round-Trip Time (RTT)

- 發送資料 -> 收到對方 ACK 的這一段時間我們稱為 RTT



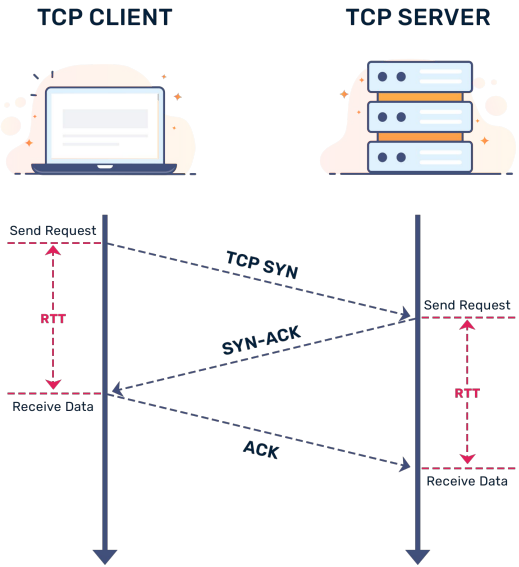
TCP - 重傳機制

- TCP 會透過特定演算法根據每一次的 RTT 動態調整重傳機制



TCP - 重傳機制

- ACK 也是有可能發生掉包的！



HTTP vs. HTTPS

HTTP vs. HTTPs

- 大家有發現某些網站瀏覽時會先跳出一個不安全 ㄇ ？
- 究竟是什麼不安全？

HTTP (Hypertext Transfer Protocol)

- 超文本傳輸協定
- 瀏覽網頁的時，網頁的內容就是透過此協定傳送
- 作用於應用層 (第 7 層，最上面)

HTTP (Hypertext Transfer Protocol)

- HTTP 不安全？
  - HTTP 跟 HTTPs 最大的差異就是「加密」
  - HTTP 相當於寄信的時候使用透明的信封
    - 運送的過程資料是完全赤裸裸的！

HTTPs

- HTTPs 多了一個關鍵的加密機制: SSL/TLS！

SSL/TLS

SSL/TLS

- SSL (Secure Sockets Layer)
- TLS (Transport Layer Security)
- TLS 是 SSL 所延伸出來的安全機制

SSL/TLS 的目標

- 加密：確保資料在傳輸過程中的機密性
- 身份驗證：確保目前網站是可信任的，非釣魚網站
- 資料完整性：確保資料被接收者收到時沒有被惡意串改

SSL/TLS 加密

- 這一連串加密解密過程牽涉到對稱式與非對稱式加密
- 過程複雜，這邊大家可以先簡單理解知道會加密即可

SSL/TLS 網頁憑證

- **網頁憑證：**
  - 某些地方要合法營業，需要有政府頒發的執照
  - 網頁憑證的角色就如同這邊的執照！

SSL/TLS 網頁憑證

- 網頁憑證由誰頒發：
  - 憑證授權單位 (CA)
  - CA 就像是公家機關一樣，負責把關憑證的發放
  - 瀏覽器會預設信任的 CA
- 網頁的憑證類型也有很多種：
  - 網域驗證憑證：
    - CA 確認連上的 Domain Name 是正確的伺服器
    - CA 同時保有這一個 Domain Name 的控制權

SSL/TLS 網頁憑證

- 最常見的憑證 (DV) 基本上就保證我們連到的是正確的網站
  - CA 無法保證這個網站背後的利用是良善的！
- 但光是有這樣保護還是不足的！網路世界的攻擊手法有非常多