Filecoin 是執行在 IPFS 上的一個激勵層，是一個基於區塊鏈的分散式儲存網路，它把雲端儲存變為一個演算法市場，代幣（FIL）在這裡起到了很重要的作用。代幣是溝通資源（儲存和檢索）使用者（IPFS 使用者）和資源的提供者（Filecoin 礦工）之間的中介橋樑，Filecoin 協定擁有兩個交易市場—資料檢索和資料儲存，交易雙方在市場裡面提交自己的需求，達成交易。

IPFS 和 Filecoin 相互促進，共同成長，解決了網際網路的資料儲存和資料分發的問題，特別是對於無數的區塊鏈專案，IPFS 和 Filecoin 將作為一個基礎設施存在。這就是為什麼我們看到越來越多的區塊鏈專案採取了 IPFS 作為儲存解決方案，因為它提供了更加便宜、安全、可快速整合的儲存解決方案。

1.1.2 IPFS 的起源

全球化分散式儲存網路並不是最近幾年的新技術，其中最著名的就是 BitTorrent、Kazaa 和 Napster 這三種，至今這些系統在全世界依舊擁有上億活躍使用者。尤其是 BitTorrent 使用者端，現在 BitTorrent 網路每天依然有超過 1000 萬個節點在上傳資料。但令人遺憾的是，這些應用最初就是根據特定的需求來設計的，在這三者基礎上靈活發展更多的功能顯然很難實現。雖然在此之前學界和業界做過一些嘗試，但自始至終沒有出現一個能實現全球範圍內低延時並且完全去中心化的通用分散式檔案系統。

之所以普及進展十分緩慢，一個原因可能是目前廣泛使用的 HTTP 協定已經足夠好用。截至目前，HTTP 是已經部署的分散式檔案系統中最成功的案例。它和瀏覽器的組合是網際網路資料傳輸和展示的最佳搭檔。然而，網際網路技術的進步從未停止，甚至一直在加速。隨著網際網路的規模越來越龐大，現有技術也越來越暴露出了諸多弊端，龐大的基礎設施投資也讓新技術的普及異常困難。

但我們說，技術都有其適用的範圍，HTTP 也是如此。四大問題使得 HTTP 面臨越來越艱巨的困難：

1）**極易受到攻擊，防範攻擊成本高**。隨著 Web 服務變得越來越中心化，使用者非常依賴於少數服務供應商。HTTP 是一個脆弱的、高度中心化的、低效的、過度依賴於骨幹網的協定，中心化的伺服器極易成為攻擊的目標。目前，為了維護伺服器正常運轉，服務商不得不使用各類昂貴的安防方案，防範攻擊成本越來越高。這已經成為 HTTP 幾乎無法克服的問題。

2）**資料儲存成本高**。經過十多年網際網路的飛速發展，網際網路資料儲存量每年呈現指數級成長。2011 年全球資料總量已經達到 0.7ZB（1ZB 等於 1 萬億 GB）；2015 年，全球的資料總量為 8.6ZB；2016 年，這個數字是 16.1ZB。到 2025 年，全球資料預計將增加至驚人的 163ZB，相當於 2016 年所產生 16.1ZB 資料的 10 倍。如果我們預計儲存 4000GB（4TB）的資料，AWS 簡單儲存服務（S3）的報價是對於第 1 個 TB 每 GB 收取 0.03 美金，對於接下來的 49TB 每 GB 收取 0.0295 美金的費用，那麼每個月將花費 118.5 美金用於磁碟空間。資料量高速成長，但儲存的價格依舊高昂，這就導緻伺服器 - 使用者端架構在今後的成本將會面臨嚴峻的挑戰。

3）**資料的中心化帶來洩露風險**。服務提供商們在為使用者提供各類方便服務的同時，也儲存了大量的使用者隱私資料。這也意味著一旦資料中心產生大規模資料洩露，這將是一場數位核爆。對於個人而言，個人資料洩露，則使用者帳號面臨被盜風險，個人隱私及財產安全難以保障；對於企業而言，訊息洩露事件會導緻其在公眾中的威望和信任度下降，會直接使客戶改變原有的選擇傾向，可能會使企業失去一大批已有的或者潛在的客戶。這並不是危言聳聽，幾乎每一年都會發生重大資料庫洩露事件。2018 年 5 月，推特被曝出現安全漏洞，洩露 3.3 億使用者密碼；2017 年 11 月，美國五角大樓意外洩露自 2009 年起收錄的 18 億筆使用者資料；2016 年，LinkedIn 超 1.67 億個帳戶在黑市被公開銷售；2015 年，機鋒網被曝洩露 2300 萬使用者

資料。有興趣的讀者可以嘗試在公開密碼洩露資料庫中查詢，是否自己的常用訊息或常用密碼被洩露，但自己卻毫不知情。

4）**大規模資料儲存、傳輸和維護難**。現在逐步進入大資料時代，目前 HTTP 協定已無法滿足新技術的發展要求。如何儲存和分發 PB 級別的大資料、如何處理高清晰度的媒體流資料、如何對大規模資料進行修改和版本疊代、如何避免重要的檔案被意外遺失等問題都是阻礙 HTTP 繼續發展的大山。

IPFS 就是為解決上述問題而誕生的。它的優勢如下：

1）**下載速度快**。如圖 1-2 所示，HTTP 上的網站大多經歷了中心化至分散式架構的變遷。與 HTTP 相比，IPFS 將中心化的傳輸方式變為分散式的多點傳輸。IPFS 使用了 BitTorrent 協定作為資料傳輸的方式，使得 IPFS 系統在資料傳輸速度上大幅度提高，並且能夠節省約 60% 的網路頻寬。

2）**最佳化全球儲存**。IPFS 採用為資料塊內容建立雜湊去重的方式儲存資料，資料的儲存成本將會顯著下降。

3）**更加安全**。與現有的中心化的雲端儲存或者個人建構儲存服務相比，IPFS、Filecoin 的分散式特性與加密演算法使得資料儲存更加安全，甚至可以抵擋駭客攻擊。

4）**資料的可持續儲存**。目前的 Web 頁面平均生命週期只有 100 天，每天會有大量的網際網路資料被刪除。網際網路上的資料是人類文明的紀錄和展示，IPFS 提供了一種使網際網路資料能夠被持續儲存的方式，並且提供資料歷史版本（Git）的回溯功能。

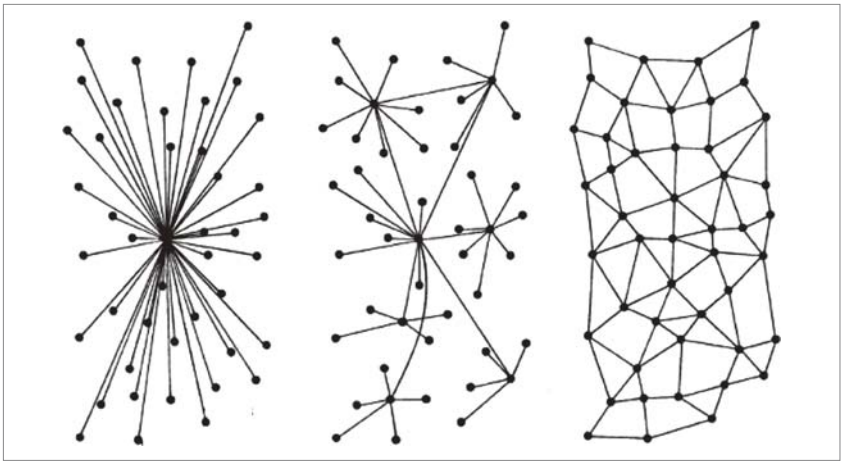

圖 1-2 中心化 - 多中心化 - 分散式技術變遷圖

上文我們提到 IPFS 技術積累已經有很多年了，它至少參考了四種技術的優點，並將它們整合在一起。這四種技術分別是分散式雜湊表 DHT、Kademlia、Git 和自驗證檔案系統（Self-Certifying File System）。

第一種對 IPFS 有借鑑意義的技術是 DHT，全稱為分散式雜湊表（Distributed Hash Table），是一種分散式儲存方法。DHT 的原理是：在不需要伺服器的情況下，每一個使用者端儲存一小部分資料，並負責一定區域的檢索，進而實現整個 DHT 網路的定址和檢索。新版 BitComet 允許同時連線 DHT 網路和 Tracker，可以在無 Tracker 的情況下進行下載。

IPFS 借鑑的第二種技術是 Kademlia。在 Kademlia 網路中，所有訊息均以雜湊表條目的形式加以儲存，這些訊息被分散地儲存在各個節點上，進而以全網構成一張巨大的分散式雜湊表。可以具體地把這張雜湊大表看成一本字典：只要知道了訊息索引的 key，便可以透過 Kademlia 協定來查詢與其對應的 value 訊息，而不管這個 value 訊息究竟是儲存在哪一個節點之上。正是這一特性確保了 IPFS 成為沒有中心調度節點的分散式系統。IPFS 還借鑑了 BitTorrent 網路。首先是消極上傳者的懲罰措施，在 BitTorrent 的使用者端上傳資料會獎勵積分，而長期不上傳的消極節點會被扣分，如果分數低於一定限度，那麼網路會拒絕

1.4 IPFS 的應用領域

IPFS 的應用領域如圖 1-11 所示。

圖 1-11 IPFS 應用領域

1. 建立長久訊息檔案

IPFS 提供了一個弱冗餘的、高效能的叢集化儲存方案。僅僅透過現有的網際網路模式來組織這個世界的訊息是遠遠不夠的，我們需要建立一個可以被世界長久記住、隨著人類歷史發展而一直存在的訊息檔案。

2. 降低儲存、頻寬成本

IPFS 提供了一個安全的點對點內容分發網路，如果你的公司業務需要分發大量的資料給使用者，IPFS 可以幫你節約大量的頻寬成本。在雲端計算時代，我們大部分的網路頻寬和網路儲存服務都由第三方服務平台來支援，例如 YouTube 這樣的大型影片平台，需要支付高額的流量費用給 ISP（網際網路服務提供商），而 YouTube 也將透過各種商業廣告及收費會員的商業形式把這部分的成本轉嫁到廣大使用者身上，整個流程體系的總成本是相當龐大的。

(2) 中心化版本控制系統

接下來人們又遇到一個問題：如何讓不同系統上的開發者協同工作？於是，中心化版本控制系統（Centralized Version Control Systems, CVCS）應運而生，如圖 2-3 所示。如 CVS、Subversion 及 Perforce 等都屬於這類系統，都有一個單一的集中管理的伺服器，儲存所有檔案的修訂版本，而協同工作的人們都透過用戶端連到這台伺服器，取出最新的檔案或者提交更新。多年以來，這已成為版本控制系統的標準做法。

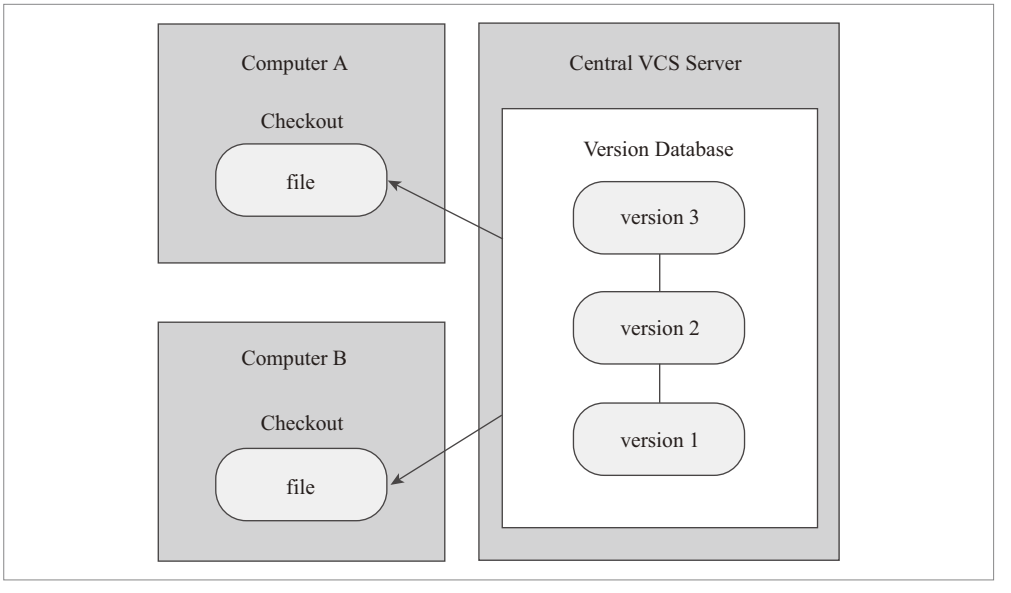

圖 2-3 中心化版本控制系統

這種做法帶來了許多好處，特別是相較於舊式的本機 VCS 有優勢。每個人都可以在一定程度上看到專案中的其他人正在做些什麼，而管理員也可以輕鬆掌控每個開發者的權限。而且管理一個 CVCS 要遠比在各個用戶端上維護本機資料庫來得輕鬆容易。這種方案最顯而易見的缺點是中央伺服器的單點故障。如果中央伺服器當機一小時，那麼在這一小時內，誰都無法提交更新，也就無法協同工作。如果中心資料庫所在的磁碟發生損壞，又沒有及時做備份，毫無疑問你將遺失所有資料，包括專案的整個變更歷史，只剩下人們在各自機器上保留

舉例來說，假設在工作目錄中有三個檔案，準備將它們暫存後提交。暫存操作會對每一個檔案計算校驗和，然後把目前版本的檔案快照儲存到 Git 倉庫中（Git 使用 blob 類型的物件儲存這些快照），並將校驗和加入暫存區域。

```
$ git add README test.rb LICENSE
$ git commit -m 'initial commit of my project'
```

當使用 git commit 建立一個提交物件前，Git 會先計算每一個子目錄（本例中就是專案根目錄）的校驗和，然後在 Git 倉庫中將這些目錄儲存為樹（tree）物件。之後 Git 建立的提交物件，除了包含相關提交訊息以外，還包含著指向這個樹物件（專案根目錄）的指標，如此它就可以在將來需要的時候，重現此次快照的內容了。

現在，Git 倉庫中有 5 個物件：3 個表示檔案快照內容的 blob 物件；1 個記錄著目錄樹內容及其中各個檔案對應 blob 物件索引的 tree 物件；以及 1 個包含指向 tree 物件（根目錄）的索引和其他提交訊息中繼資料的 commit 物件。概念上來說，倉庫中的各個物件儲存的資料和相互關係看起來如圖 2-7 所示。

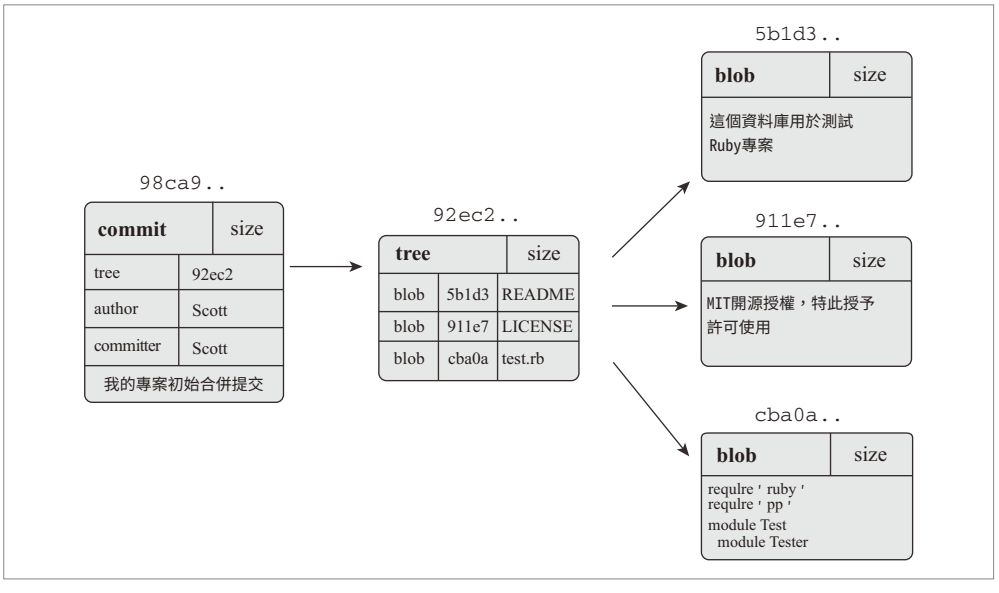

圖 2-7 單一提交物件的資料結構

做些修改後再次提交，那麼這次的提交物件會包含一個指向上次提交物件的指標。兩次提交後，倉庫歷史會變成圖 2-8 所示的樣子。

現在來談分支。Git 中的分支本質上僅是個指向 commit 物件的可變指標。Git 使用 master 作為分支的預設名字。第一個分支也常被稱為主幹。主幹可被複製為其他分支，每條分支的可變指標在每次提交時都會自動向前移動，如圖 2-9 所示。

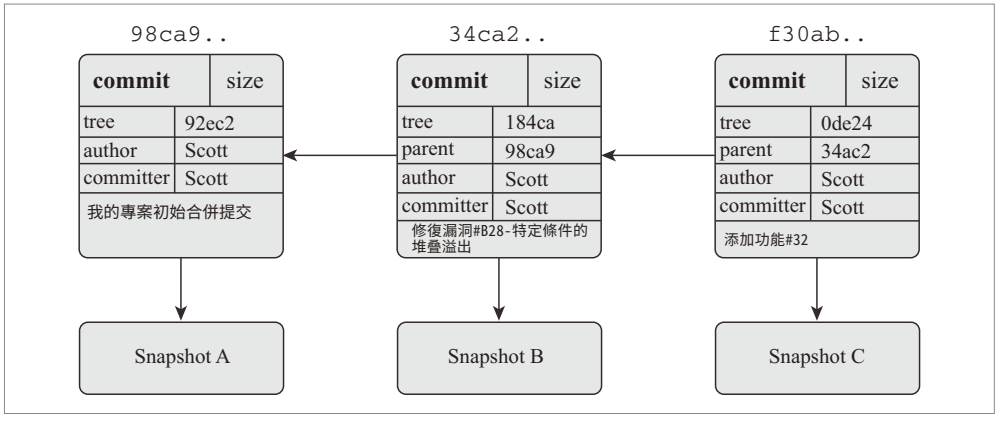

圖 2-8 多個提交物件之間的連結關係

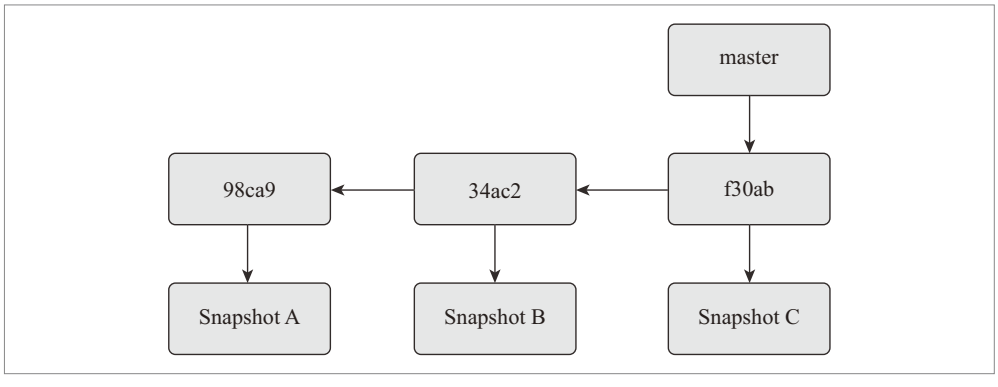

圖 2-9 分支中的歷史提交

- 可連線性：使用 ICE 等 NAT 穿越技術來實現廣域網的可連線性。
- 完整性：使用雜湊校驗檢查資料完整性，IPFS 網路中所有資料塊都具有唯一的雜湊值。
- 可驗證性：使用資料發送者的公鑰及 HMAC 訊息認證碼來檢查訊息的真實性。

IPFS 幾乎可以使用任意網路進行節點之間的通訊，沒有完全依賴於 IP 協定。IPFS 透過 multiaddr 的格式來表示目標位址和其使用的協定，以此來相容和擴展未來可能出現的其他網路協定。

```
# an SCTP/IPv4 connection
/ip4/10.20.30.40/sctp/1234/
# an SCTP/IPv4 connection proxied over TCP/IPv4
/ip4/5.6.7.8/tcp/5678/ip4/1.2.3.4/sctp/1234/
```

IPFS 的網路通訊模式是遵循覆蓋網路（Overlay Network）的理念設計的。覆蓋網路的模型如圖 3-2 所示，是一種網路架構上疊加的虛擬化技術模式，它建立在已有網路上的虛擬網路，由邏輯節點和邏輯鏈路構成。圖中多個容器在跨主機通訊時，使用 Overlay Network 網路模式。首先虛擬出類似服務匣道的 IP 位址，例如 10.0.9.3，然後把封包轉發到 Host（主機）物理伺服器位址，最終透過路由和交換到達另一個 Host 伺服器的 IP 位址。

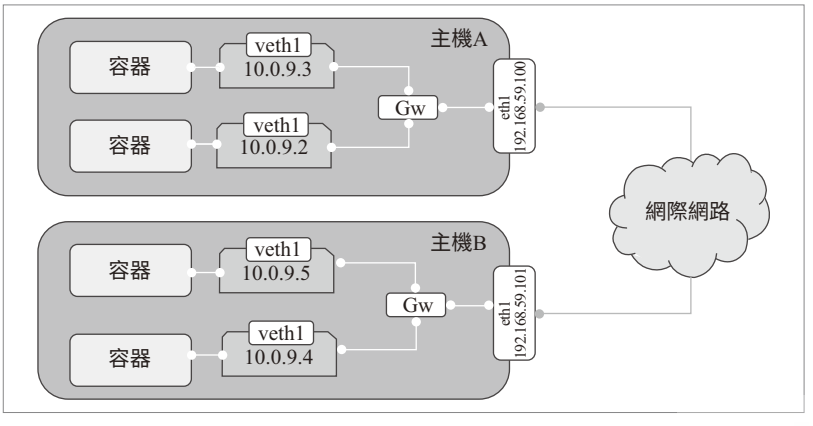

圖 3-2 覆蓋網路模型

```
type Peer struct {
    nodeid NodeId
    ledger Ledger // 對等節點之間的分類帳單
    last_seen Timestamp // 最後收到訊息的時間戳
    want_list []Multihash // 需要的所有區塊校驗
}
// 協定介面：
interface Peer {
    open (nodeid : NodeId, ledger : Ledger);
    send_want_list (want_list : WantList);
    send_block(block: Block) -> (complete:Bool);
    close(final: Bool);
}
```

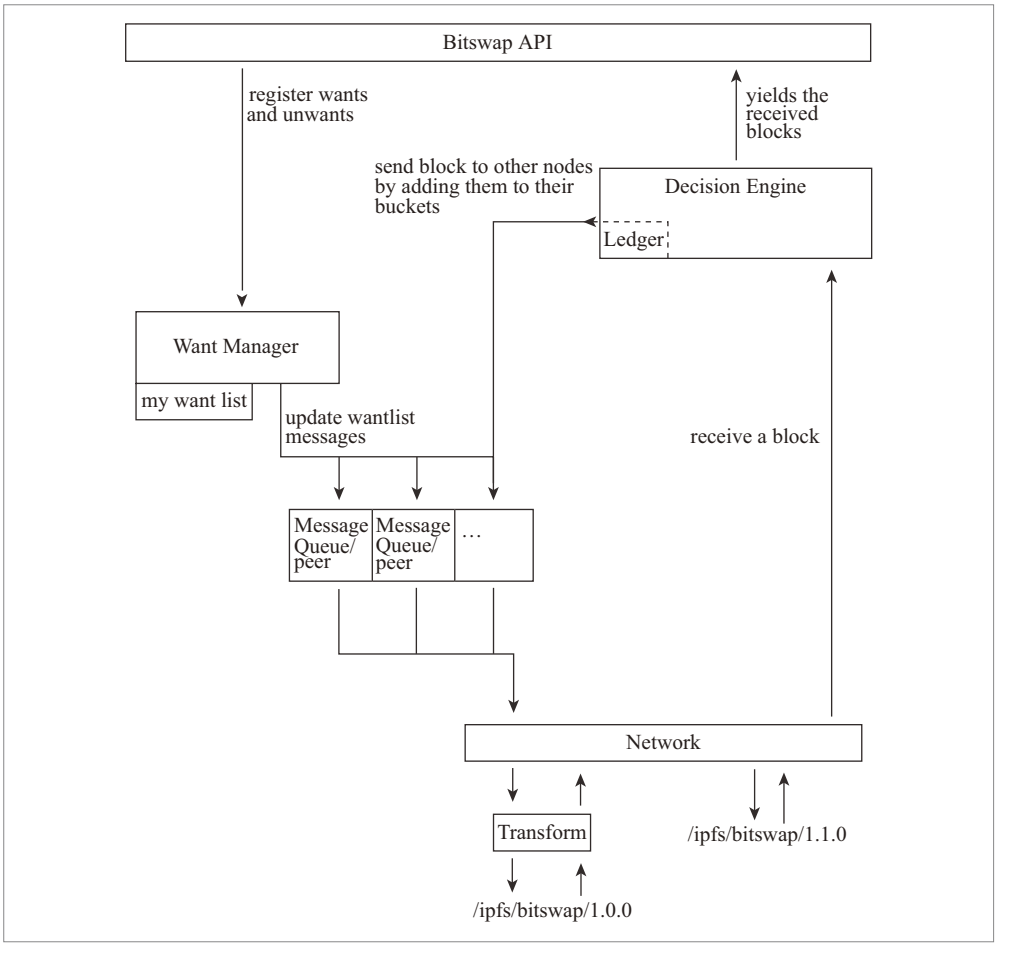

圖 3-3 BitSwap 協定流程

```
            "name": "ttt333-name" },
            { "hash": "<lll111-hash>", "size": 587,
            "name": "ttt333-name/lll111-name"},
            { "hash": "<bbb222-hash>", "size": 22,
            "name": "ttt333-name/lll111-name/bbb222-name" },
            { "hash": "<bbb222-hash>", "size": 22
            "name": "bbb222-name" }
        ]
    }
```

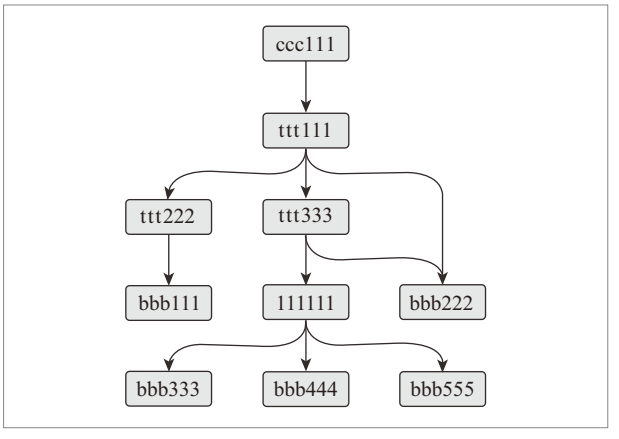

圖 3-4 物件關係圖範例

3.7 命名層（Naming）

3.7.1 IPNS：命名以及易變狀態

IPFS 形成了一個內容可定址的 DAG 物件，我們可以在 IPFS 網路中發布不可更改的資料，甚至可以跟蹤這些物件的版本歷史記錄。但是，存在一個很嚴重的問題：當資料物件的內容更新後，同時發生改變的還有內容位址的名稱。我們需要一種能在易變環境中保持固定名稱的方案，為此，協定實驗室團隊為 IPFS 設計了 IPNS 星際檔案命名系統模組。

(5) swarm

swarm 是 libp2p 的核心元件之一，因為它是真正的網路層，如圖 4-8 所示。所有與網路相關的元件全部位於 swarm 元件裡面，位址簿、連結、監聽器等元件都在這裡進行管理。swarm 有回調機制，當有一個新的 stream 進來，呼叫中轉函數進行邏輯處理。transport 管理功能可對多種 transport 管理，以實現更靈活的功能。dialer 是撥號器，它包含三種撥號器：同步撥號器、後台限制撥號器和有限制的撥號器。三種撥號器共同完成整個撥號的過程。

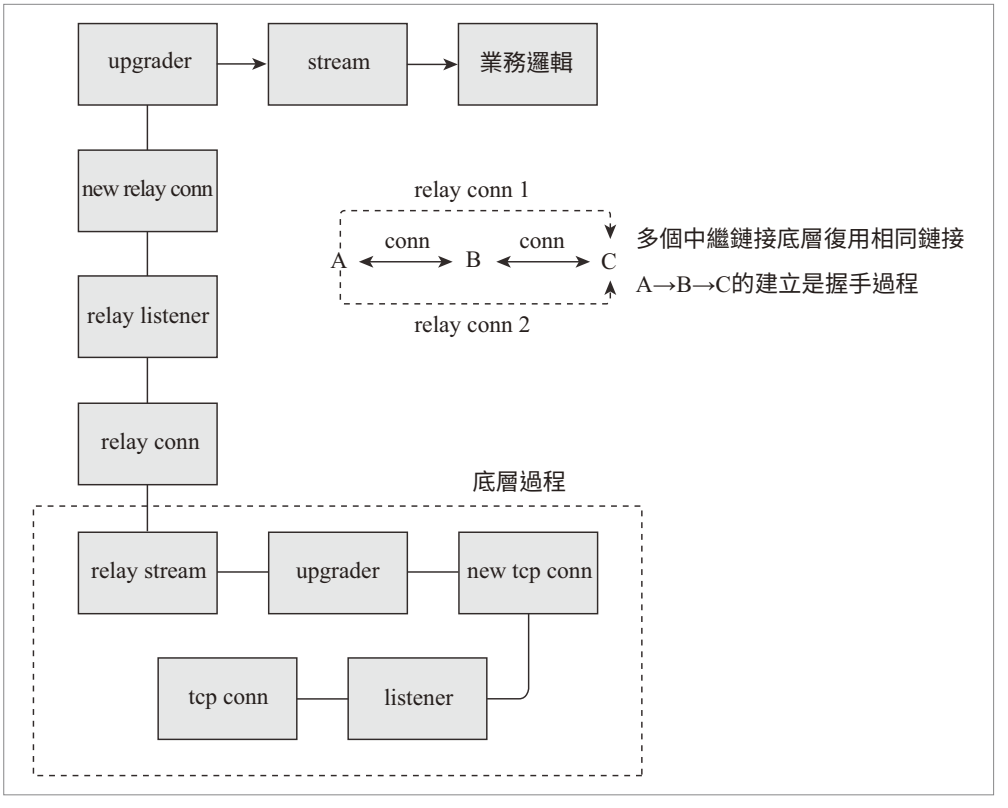

圖 4-6 libp2p 中繼方案實現方式