國立交通大學

資訊管理研究所

碩士論文

一個以 Binary Tree 架構為基礎的 RFID 辨識機制

A Novel Binary-Tree-Based RFID Identification Scheme

研究生：劉宇哲

指導教授：羅濟群博士

中華民國九十四年六月

一個以 Binary Tree 架構為基礎的 RFID 辨識機制

A Novel Binary-Tree-Based RFID Identification Scheme

研 究 生：劉 宇 哲　　Student: Yu-Che Liu

指導教授：羅 濟 群　　Advisor: Chi-Chun Lo

國立交通大學

資訊管理研究所

碩士論文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Business Administration

in

Information Management

June 2005

Hsinchu, Taiwan, the Republic of China

中華民國九十四年六月

一個以 Binary Tree 架構為基礎的 RFID 辨識機制

研究生：劉宇哲　　指導教授：羅濟群 老師

國立交通大學資訊管理研究所

摘要

RFID 是近年來所發展出來的一套無線射頻辨識系統，它被廣泛地使用在物流作業上，並且漸漸地取代傳統的 Bar Code 來辨識貨品，透過被附著在貨品上的 RFID 標籤，利用無線電波的傳輸方式，RFID 標籤的 ID Code 被 Reader 所讀取。在 RFID 的標準中，標籤和 Reader 的辨識方式主要是以 Binary Tree 辨識模式為基礎，然而 Binary Tree 辨識模式在標籤數量較多的環境下，辨識過程較為冗長，所需花費的時間較多，主要的原因是當標籤數量增多的時候 Reader 和標籤的總溝通次數會快速地增加，如果在多標籤的辨識環境下，減少標籤和 Reader 之間的溝通次數將是最重要的課題。在此，本論文嘗試採用一種以 Binary Tree 辨識模式為基礎的新式辨識機制，來試圖減少原本在 Binary Tree 辨識模式中 Reader 和標籤在辨識過程中總共所需的溝通次數，以藉此縮短 Reader 在辨識過程中所需耗費的時間以及所需的電力。

模擬的過程中，主要是根據標籤的長度、標籤的數量、標籤 ID Code 的相似度以及使用的記憶體數量這四個變因來分析原本的 Binary Tree 辨識模式和本論文所提出的辨識機制在標籤的總回應次數和 Reader 的總查詢次數的差異，以及改善的程度比例。由模擬實驗結果觀之，在標籤位元長度愈長、標籤數量愈多以及標籤 ID Code 的相似度愈高的情況下，本論文所提出的辨識機制能夠有效地降低辨識過程中的總溝通次數，而降低的幅度平均大約在十五到六十個百分點左右。

關鍵字：RFID、Binary Tree 辨識模式、溝通次數

A Novel Binary-Tree-Based RFID Identification Scheme

Student：Yu-Che Liu　　Advisor：DR. Chi-Chun Lo

Institute of Information Management
Nation Chiao Tung University

Abstract

RFID is one kind of identification system developed in recent years. It was used in the distribution operation broadly and replaced traditional Bar Code to identify products gradually. The RFID tags attached to products are read by RFID reader using radio transmission.

According to the standard of RFID，the identification mode between the tags and reader is based on Binary Tree Identification Scheme mainly. However，the process of Binary Tree Identification Scheme is copious when the tags are numerous , and it takes much time. The root cause is that the communication times between the reader and tags will increase rapidly when the tags are more and more. To decrease the communication times between the reader and tags will be the most important course in the multi-tags environments.

According above，this paper attempted to propose a novel identification scheme based on Binary Tree Identification Scheme to reduce the total communication times between the reader and tags. So that it can shorten the time and save the power in the identification process.

In the simulation process，this paper analyzed the difference of the tag response times、reader inquiring times and improved ratio between the BT scheme and improved scheme according to four aspects：tag bit-length、the number of tag、the similarity of tags' ID Code and the amount of memory used. According to the simulation results，the novel scheme this paper proposed can reduce the total communication times. And the average decrease of the communication times reaches fifteen to sixty percentages.

Keyword：RFID、Binary Tree Identification Scheme、communication times

誌謝

能夠順利完成論文並從交大資管所畢業，首先要感謝羅老師在每次Meeting的時候能給予我許多的指導，讓我能夠更順利地完成論文。除此之外，也同時要感謝我的口試老師陳瑞順老師以及陳文賢，在口試時所給予我的寶貴意見。

更要感謝明橋哥哥的大力幫忙、不良的關懷，大雄的免費專車，小雞的日常開導，還有笨毛常常找我和狗狗喝酒，這些都讓我的碩士生涯增添許多快樂。

也感謝在網路實驗室的眾多夥伴，仁德、鼎元、明諺在這兩年內給予我的許多幫助，俊龍、俊傑學長在多方面的支援，還有家安的搞笑演出更是讓我的碩士生涯生色不少，也感謝碩一學弟所帶來的歡笑。

最後，當然要感謝我的家人，讓我能沒有後顧之憂，感謝先草密的土地公讓我有求必應。

目次

圖目錄

表目錄

第一章　緒論

1.1　研究背景與動機

Radio Frequency Identification (RFID)[2]是近年來新崛起的無線辨識技術，它是在西元 1998 年的時候，由 Sanjay Sarma 和 David Brock[1]最先提出的自動化辨識系統技術，並且在西元 1999 年於美國 MIT 成立的 Auto-ID Center 主導其技術的發展，而目前 Auto-ID Center 已經將相關的技術移轉至 EPC Global，並繼續從事發展相關的技術和標準的制定工作。

在 RFID 辨識技術出現之前，主要都是以 Bar Code[2]作為主要的辨識方式，然而，RFID 的辨識技術卻在許多方面比 Bar Code 更具優勢，像是 RFID 標籤識別碼是獨一無二的、Reader 一次可以讀取一個以上的標籤、Reader 和標籤的距離可以拉的比較遠和 RFID 標籤可以重用等優點，讓 RFID 的辨識技術能漸漸地取代 Bar Code。

RFID 的辨識方式是以 Binary Tree 辨識演算法[7]為基礎，而 Binary Tree 辨識演算法是以一次辨識一個 Bit 為原則，每次 Reader 廣播一個 Bit 值給所有的標籤，標籤收到這個位元值後就和自身的 ID Code 相對應位元作比對，如果相同就回應給 Reader 知道，如果不同就進入 Sleep Mode，這樣的動作持續到辨識出一個標籤後，Reader 就完成了一個回合的查詢，一直重複這樣的回合查詢，直到所有的標籤都被 Reader 所讀取為止。

Binary Tree 的辨識模式在當同時要被辨識的標籤數量愈多的時候，Reader 和標籤的總溝通次數會很明顯地急劇性增加，這使得 Binary Tree 辨識模式在標籤數量較多的時候顯得效率不彰，造成這樣的結果主要是因為 Binary Tree 的辨識模式會造成許多的溝通步驟一直被重複進行，而這種情形在當標籤數量愈多的時候愈明顯。

在本論文所提出的辨識機制中，在 Reader 端增加一些少許的記憶體後，Reader 將先前得到的查詢結果記憶下來，以作為日後 Reader 查詢的根據，透過這樣的方式可以有效地減少 Reader 和標籤之間的總溝通次數，進而增進辨識流程的效率。

1.2 研究目的

本篇論文的目的在於提出一個以 Binary Tree 辨識架構為基礎的新辨識機制，讓 RFID Reader 在多標籤需要同時一起被辨識的環境下時，能夠有效地減少在辨識過程中，Reader 和標籤的總溝通次數，這個部份包含了 Reader 所送出的查詢次數，以及標籤的回應次數，讓辨識過程所需的時間可以縮短，使效率得以提升。

1.3 章節說明

在第二章裡，本論文將會先說明 RFID 的基本原理和辨識技術，以及辨識的流程如何運作；接下來再繼續介紹一些基本的辨識架構。第三章的部分，主要是介紹本論文所提出的辨識機制運作基本原理，以及一些例子的解說。第四章的部分則是說明模擬實驗的系統設計、各功能模組、測試環境以及模擬實驗內容以及根據模擬實驗的結果做出一些分析和結論。第五章則是未來發展與研究方向的部分，探討還有哪些可以改進或加強的地方，最後則是相關的參考文獻部分。

第二章　文獻探討

在這一個章節裡，本論文將會先介紹 RFID 的基本觀念以及 RFID 的各種標準，接下來將繼續介紹兩種標籤辨識演算法 Binary Tree Algorithm 和 Query Tree Algorithm，之後再各以一個例子來解說這兩個演算法的流程，並在本章節的最後討論這兩者之間的優缺點比較。

2.1　RFID

RFID 是 Radio Frequency Identification 無線射頻辨識的縮寫[1]，透過一個 Reader 和數個 Tag 在無線電波中傳送 0 和 1 的 Bits，並藉此達成 Tag ID 的辨識。由於 RFID 能夠提供一個廉價且自動化的辨識系統，再加上他能夠和 Internet 輕易地做結合，因此，RFID 常常被應用在現今大部分的 ID 辨識系統中，尤其在物流業甚至是在企業的整個供應鍊上，RFID 的辨識系統都扮演著非常重要的角色。

大約在二十年前左右，條碼 (Bar Code) [2]是當時最主要的 ID 辨識系統，並且大量出現在物流管理的作業上，但是到了最近幾年，條碼的辨識方式已經漸漸被 RFID 的 Reader 和 Tag 所取代，而主要的原因是因為現今的條碼辨識方式面臨了以下的問題：

- ✧ 條碼的辨識方式面需要有作業員在旁操作，無法全自動化。
- ✧ 一次只能辨識一個條碼，當 Tag 數量較多時，效率較差。
- ✧ 在辨識條碼時，被辨識的條碼必須離 Reader 很近(數公分)。
- ✧ 條碼為單向的通信，無法提供及時訊息且難以與網際網路相連接。
- ✧ 條碼的 ID 值無法改變，重用性低。

圖 1　Bar Code 範例

相較於上述條碼的缺點，RFID 的辨識技術在各方面都有相對的優勢，首先，在 RFID 的辨識過程中，由於透過 Radio 的傳輸方式，使得 Reader 和 Tag 之間的距離可以拉長許多，所以也就不需要像條碼必須有作業員把 Tag 擺到 Reader 非常有限的讀取範圍內，所以 RFID 能夠把辨識的過程全自動化。再者，由於 RFID Reader 和 Tag 之間的辨識距離較為寬鬆，所以可以一次辨識超過一個 Tag。另外 RFID 的標籤為可程式化的標籤，所被賦予的 ID 值在日後可以做變更，而重新代表另一個產品的 ID，所以重用性非常高。

基於以上的種種原因，在現今越來越以全自動化以及低成本的商業模式下，條碼的 ID 辨識技術漸漸地失去它的優勢，而正由於 RFID 的上述種種優勢，而導致了 RFID 能夠漸漸地取代 Bar Code 的辨識模式，而逐漸成為主要的主流趨勢。

2.1.1　RFID 的標準化

大約在西元 1999 年，美國麻省理工學院的兩位教授，Sanjay Sarma 和 David Brock 開始將 RFID 的技術標準化[1]，而提出 Auto-ID 的架構，並成立了自動識別技術中心(AUTOID CENTER)，提出了 EPC 概念，其後四個世界著名研究性大學:英國劍橋大學、澳大利亞的阿雷德大學、日本 Keio 大學、上海復旦大學相繼加入參與研發 EPC，並得到了 100 多個國際大公司的支援，其研究成果已在一些公司如寶潔公司、Tesco 公司中試用。2004 年 10 月份，EAN.UCC 正式接管了 EPC 在全球的推廣應用工作，成立了 EPC Global。而

Auto-ID Center 改為 Auto-ID Lab，EPC 的研究性工作也將繼續由 Auto-ID Lab 承擔。作為 EAN.UCC 的會員組織，中國物品編碼中心（ANCC）也積極參與到 EPC 的推廣工作中來。

由於 Auto-ID Tag 的極低成本以及 Auto-ID 本身已是公開且被大家所廣泛接受的標準，因此常被業界拿來作為物流作業中管理物品的工作。Auto-ID Tag 被附著在箱子、盒子或是任何產品之上，在供應鍊管理系統中提供管理者雙向的資訊流。

Auto-ID 其實是一套整合的物件資訊取得方式，而整個 Auto-ID 的架構包含[1]：

- RFID Tag
- RFID Reader
- 標準：
  - 物件本身的產品是別代碼 (Electronic Product Code , EPC)
  - 物件本身的資訊描述語言 (Product Markup Language, PML)
  - 識別碼解析服務 (Object Name Service, ONS)
- 架構：
  - 儲存物件資訊的伺服器 (PML Server)
  - 查詢物件資訊所在的伺服器 (ONS Server)
  - Savant：將 EPC Code 送至 ONS Server 以尋找物件的 PML 檔案

而整個 Auto-ID 的運作架構大致可以歸納如下圖：

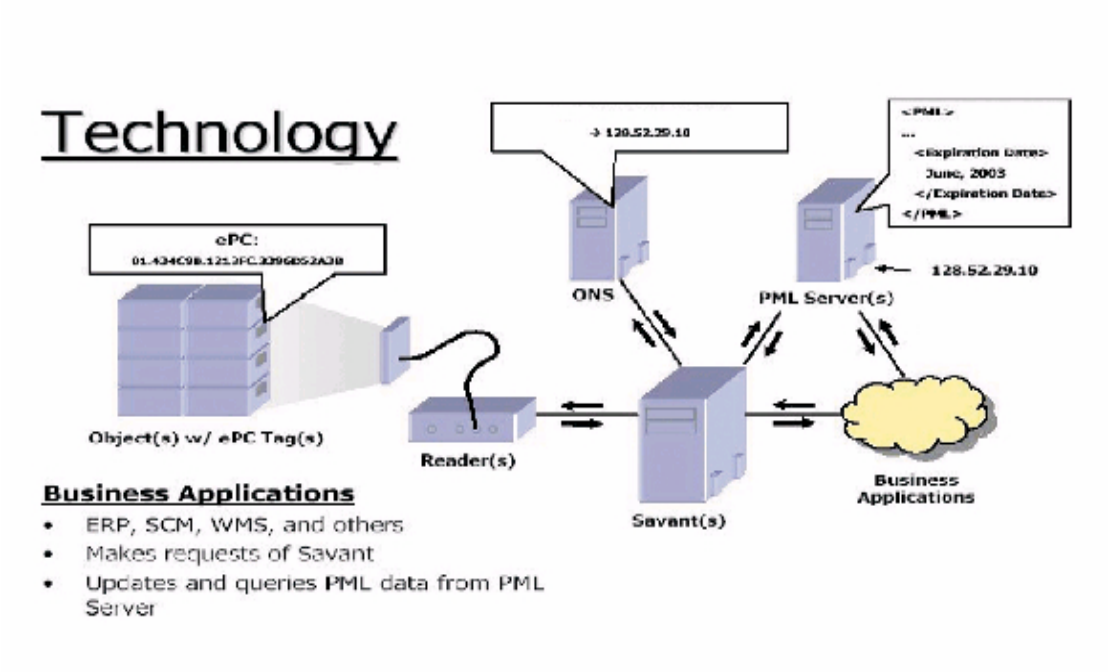

圖 2　Auto-ID 流程圖[1]

- **EPC (Electronic Product Code)[1]**

由於產品的唯一識別對於某些商品非常必要，而條碼識別最大的缺點之一是它只能識別一類產品，而不是唯一的商品。例如牛奶紙盒上的條碼到處都一樣，要辨別哪盒牛奶會先超過有效期將是不可能的。那麼如何才能識別和跟蹤供應鏈上的每一件單品呢？目前所找到的最好的解決方法就是給每一個商品提供唯一的號碼--"EPC 碼"。

EPC 碼是新一代的與 EAN/UPC 碼相容的新的編碼標準，在 EPC 系統中 EPC 編碼與現行 GTIN 相結合，因而 EPC 並不是取代現行的條碼標準，而是由現行的條碼標準逐漸過渡到 EPC 標準或者是在未來的供應鏈中 EPC 和 EAN.UCC 系統共存。

EPC 碼是由一個版本號加上另外三段資料（依次為功能變數名稱管理者、物件分類、序列號）組成的一組數位。其中版本號標識 EPC 的版本號，它使得 EPC 隨後的碼段可以有不同的長度；功能變數名稱管理是描述與此 EPC 相關的生產廠商的資訊，例如"可口可樂公司"；物件分類記錄產品精確類型的資訊，

例如："美國生產的 330ml 罐裝減肥可樂（可口可樂的一種新產品）"；序列號唯一標識貨品，它會精確的告訴我們所說的究竟是哪一罐 330ml 罐裝減肥可樂。下圖便是一個 96bits 的 EPC 識別碼。

圖 3　96bits EPC 識別碼格式[3]

一般的辨識標籤依其不同的分類標準可以大致分為[3]：

- 依能量傳遞方式而分為主動式標籤與被動式標籤。
- 根據射頻標籤的讀寫方式可以分為只讀型標籤和讀寫型標籤。
- 根據射頻標籤有無電源可分為無源標籤和有源標籤。
- 根據射頻標籤的工作頻率可分為低頻標籤、高頻標籤和微波標籤。
- 根據射頻標籤的工作距離可分為遠程標籤、近程標籤、超近程標籤。

而 EPC 標籤的特色包含：

- EPC 標籤中儲存的唯一資料是 96 位或者 64 位產品電子代碼，
- EPC 標籤通常是被動式射頻標籤
- EPC 標籤是全球統一標準、規格化的射頻標籤，其相關技術特性，如資訊儲存格式、與讀碼器間的通信協定，包括無線頻率、數據通信方式等統一於 EPC 系統標準。

EPC 識別碼主要有幾種不同的類別，我們以下表說明之：

表 1　EPC Code 類別

		版本號	功能變數名稱管理	對象分類	序列號
EPC-64	TYPE Ⅰ	2	21	17	24
	TYPE Ⅱ	2	15	13	32
	TYPE Ⅲ	2	26	13	23
EPC-96	TYPE Ⅰ	8	28	24	36
EPC-256	TYPE Ⅰ	8	32	56	160
	TYPE Ⅱ	8	64	56	128
	TYPE Ⅲ	8	128	56	64

- **Savant [1]**

每件產品都加上 RFID 標籤之後，在產品的生產、運輸和銷售過程中，Reader 將不斷收到一連串的 EPC 碼。整個過程中最為重要、 同時也是最困難的環節就是傳送和管理這些資料，於是 EPC Global 開發了一種名叫 Savant 的軟體技術。Savant 拿到從 Reader 送來的 EPC 辨識碼之後，Savant 便會把 EPC 辨識碼當成輸入，送到 ONS 以查詢此項 EPC 辨識碼所代表的產品種類，以及產品資訊。

歸納其主要任務有：

（1） 資料校對：

處在網絡邊緣的 Savant 系統，直接與讀碼器進行資訊交流，它們會進行資料校對，並非每個標籤每次都會被讀到，而且有時一個標籤的資訊可能被誤讀，Savant 系統能夠利用運算法校正這些錯誤。

（2） 讀碼器間協調：

如果從兩個有重疊區域的讀碼器讀取信號，它們可能讀取了同一個標籤的資料，產生了相同多餘的產品電子代碼，Savant 的一個任務就是分析已讀取的資訊並且刪掉這些冗餘的產品代碼。

（3） 資料傳送：

在一個層次上，Savant 系統必須決定什麼樣的資訊需要在供應鏈上向上傳遞或向下傳遞，例如，在冷藏工廠的 Savant 系統可能只需要傳送它所儲存的商品溫度資訊就可以了。

（4） 資料儲存：

現在的資料庫不具備在一秒鐘內處理超過幾百條事件的能力，因此 Savant 系統的另一個任務就是維護實時儲存事件資料庫，本質上來講，系統取得實時產生的產品電子代碼並且智能地將資料儲存，以便其他企業管理的應用程序有權訪問這些資訊，並保證資料庫不會超負荷運轉。

（5） 任務管理：

無論 Savant 系統在層次結構中所處的等級是什麼，所有的 Savant 系統都有一套獨具特色的任務管理系統(TMS)，這個系統使得它們可以實現用戶自己定義的任務來進行資料管理和監控，例如，一個商店中的 Savant 系統可以通過編寫程序實現一些功能，當貨架上的產品降低到一定水平時，會給倉庫管理員發出警報。

- **識別碼解析服務 (Object Name Service, ONS) [1]**

Auto-ID 中心認為一個開放式的，全球性的追蹤物品的網路需要一些特殊的網路結構。因為除了將 EPC 碼存儲在標籤中外，還需要一些將 EPC 碼與相應商品資訊進行匹配的方法。這個功能就由物件名解析服務(ONS)來實現，它是一個自動的網路服務系統，類似於功能變數名稱解析服務(DNS)，DNS 是將一台電腦定位到網上的某一具體地點的服務。

當一個解讀器讀取一個 EPC 標籤的信息時，EPC 碼就傳遞給了 Savant 系統。Savant 系統然後再在局域網或網際網路上利用 ONS 物件名解析服務找到這個產品資訊所存儲的位置。ONS 給 Savant 系統指明了存儲這個產品的有關資訊的 PML 伺服器，因此就能夠在 Savant 系統中找到這個檔，並且將這個檔中的

關於這個產品的資訊傳遞過來，從而應用於供應鏈的管理。整個 ONS 的查詢步驟如下：

1. 從標籤上判讀一個 EPC 代碼資料字串
2. Reader 將此 EPC 代碼字串發送到本地伺服器
3. 本地伺服器對 EPC 代碼資料進行適當排列，過濾，將 EPC 代碼發送到本地 ONS 伺服器
4. 本地 ONS 伺服器利用格式化轉換字符串將 EPC 比特位編碼轉變成 EPC 域前綴名，再將 EPC 域前綴名與 EPC 域后綴名結合成一個完整的 EPC 域名，ONS 伺服器再進行一次 ONS 查詢，將 EPC 域名發送到指定 ONS 伺服器基礎架構，以獲得所需要的資訊
5. ONS 基礎架構給本地 ONS 伺服器回送 EPC 域名對應一個或多個 PML 伺服器 IP 位址
6. 本地 ONS 運算器再將 IP 位址回送給本地伺服器
7. 本地伺服器再根據 IP 位址聯繫正確的 PML 伺服器，以獲取所需的 EPC 資訊

ONS 運作過程和步驟，用示意圖說明如下：

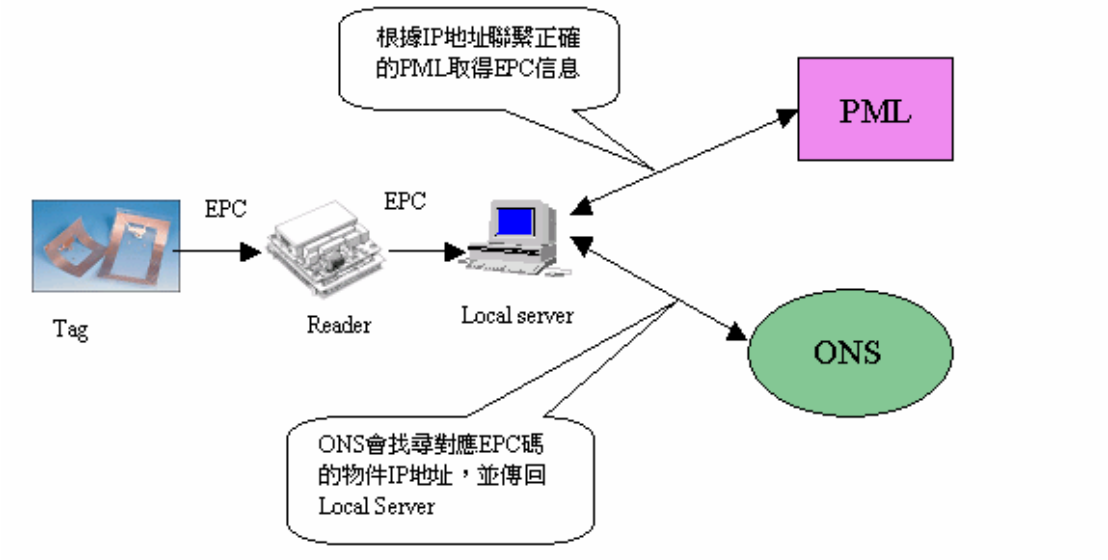

圖 4　ONS 查詢流程[2]

- **物件資訊描述語言(Product PML) [1]**

EPC Code 用來表示單一物件的識別碼，而所有關於產品有用的資訊都用一種新型的標準的電腦語言---物件資訊描述語言（PML）所撰寫，PML 是基於為人們廣為接受的可擴展標識語言（XML）發展而來的。因為它將會成為描述所有自然物體、過程和環境的統一標準，PML 的應用將會非常廣泛，並且進入到所有行業。AUTO-ID 中心的目標就是以一種簡單的語言開始，鼓勵採用新技術。PML 還會不斷發展演變，就像網際網路的基本語言 HTML 一樣，演變為更複雜的一種語言。

PML 將提供一種通用的方法來描述自然物體，它將是一個廣泛的層次結構。例如，一罐可口可樂可以被描述為碳酸飲料，它屬於軟飲料的一個子類，而軟飲料又在食品大類下面。當然，並不是所有的分類都如此的簡單，為了確保 PML 得到廣泛的接受，AUTO-ID 中心依賴於標準化組織已經做了大量工作，比如國際重量度量局和美國國家標準和技術協會等標準化組織制定的相關標準。

除了那些不會改變的產品資訊（如物質成分）之外，PML 將包括經常性變動的資料（動態資料）和隨時間變動的資料（時序資料）。在 PML 檔中的動態資料包括船運的水果的溫度，或者一個機器震動的級別。時序資料在整個物品的生命週期中，離散且間歇地變化，一個典型的例子就是物品所處的地點。通過使所有這些資訊通過 PML 檔都可得到，公司將能夠以新的方法利用這些資料。例如，公司可以設置一個觸發器，以便當有效期將要結束時，降低產品的價格。

PML 是以 XML 為基礎所發展出來的新的標準電腦網路語言，為一通用標準， PML 的目標是為物理實體的遠程監控和環境監控提供一種簡單、通用的描述語言，可廣泛應用在存貨跟蹤、自動處理事務、供應鏈管理、機器控制和物對物通訊等方面。

PML 檔案將被存儲在一個 PML 伺服器上，此 PML 伺服器將配置一個專用的電腦，為其他電腦提供他們需要的檔。PML 伺服器將由製造商維護，並且儲存這個製造商生產的所有商品的資訊。

下面是一個 PML 檔案的例子，記錄了某個 EPC Code 所對應的日期資訊，而由於 PML 描述語言是從 XML 演變而成的語言，所以在這個例子中，我們不難發現，PML 描述語言的語法以及定義方式都和 XML 非常相似，也正是因為這樣，PML 描述語言能夠和 XML 相容，而能更容易透過 TCP/IP 在網際網路上傳遞 PML 檔案裡所包含的資訊。

```
<pmlcore: Sensor>
    <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
    <pmlcore:Observation>
        <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
        <pmlcore:Tag>
            <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
        </pmlcore:Tag>
        <pmlcore:Tag>
            <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
        </pmlcore:Tag>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

圖 5　PML 檔案範例

2.1.2　實例解說

RFID 的辨識技術最常被應用在物流管理的作業上，現在我們就用一個貨物進出的管理實例來詳細解說 RFID 如何運用在物流作業上。

1. 在每一個物品上加上標籤，此例為可樂罐，並且在裝箱的時候也在箱子外加上 RFID 的標籤。在箱子上附加 RFID 的標籤是為了在運送的過程中，也能予以識別。

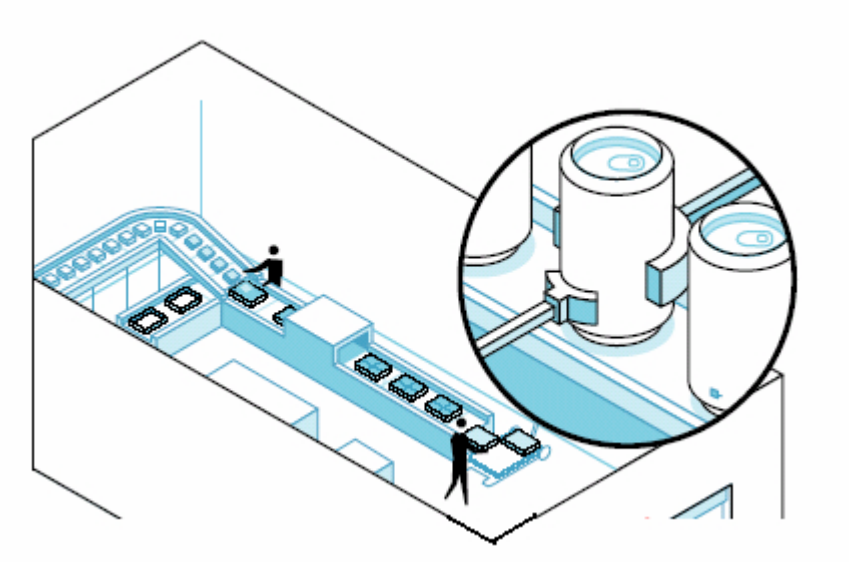

圖 6　在貨品上附著 RFID 標籤[2]

2. 可樂出貨時，在出貨站門口所裝置的 Reader 把每一箱裡的標籤都一一讀入並把貨物裝上貨車。

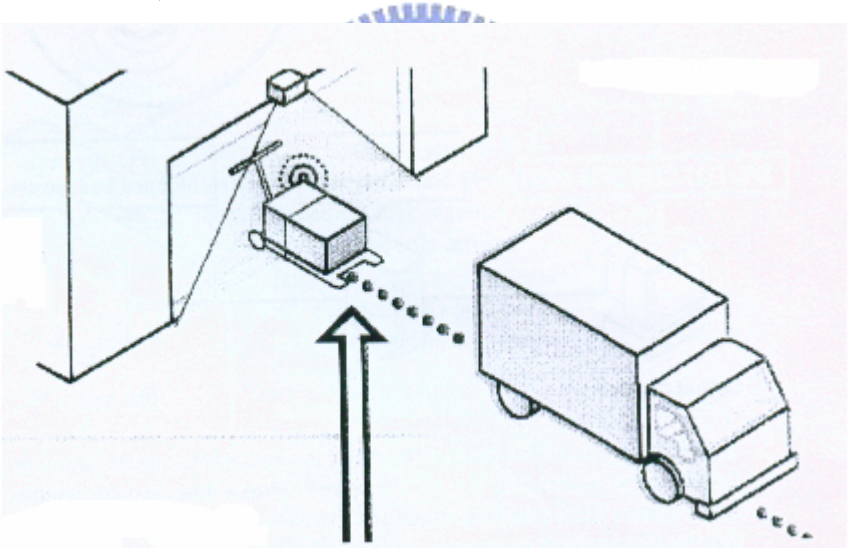

圖 7　讀取貨品上的 RFID 標籤[2]

3. Reader 將所收集的 EPC Code 傳送給 Savant，隨後 Savant 根據所收集的 EPC Code 向 ONS 發出詢問，ONS 在比對完 Savant 送來的 EPC Code 後找出貨品額外資訊 PML Server 的位置。此時 Savant 便可到 PML Server 找到貨品的額外資訊。

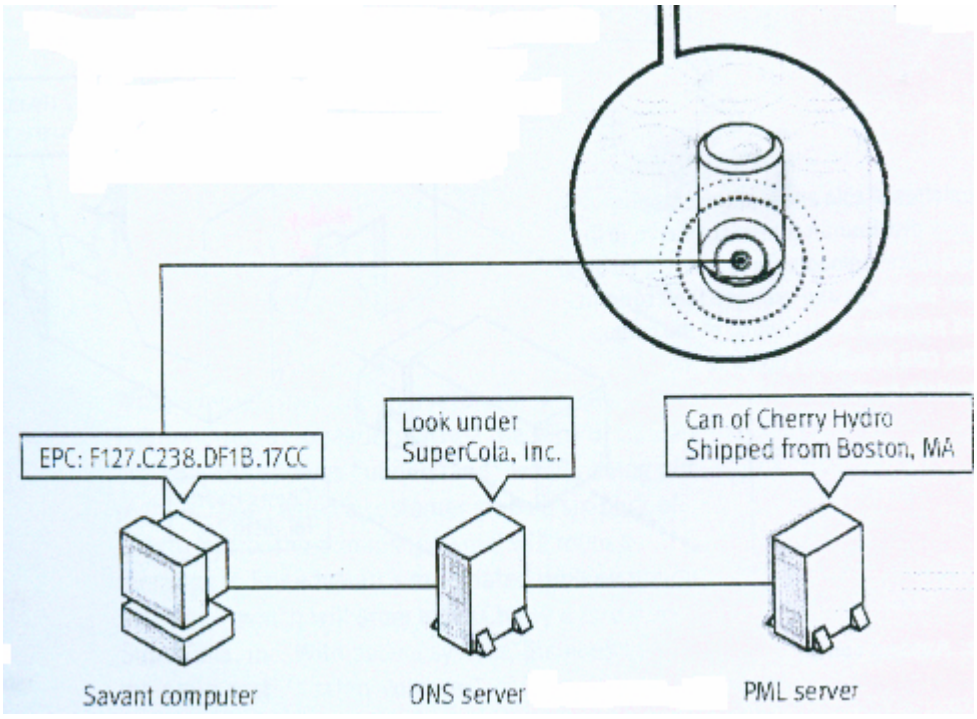

圖 8　將標籤資訊傳遞給各個伺服器[2]

4. 送達配銷中心後，透過配銷中心的 Reader 可以明確地知道哪些貨品已經到貨，而不用拆箱驗收，最後根據貨品的資訊將貨品裝上適當的運貨車，再由運貨車送往目的地。

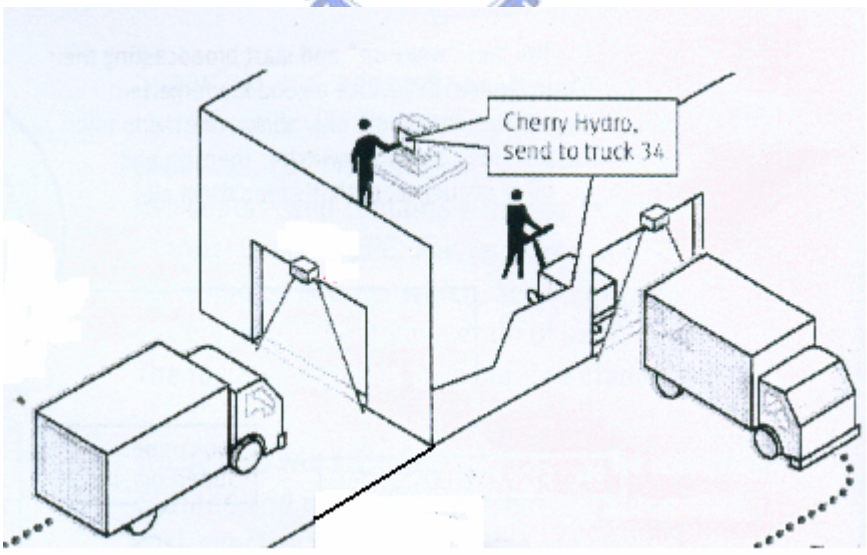

圖 9　送達目的地[2]

5. 當貨物送到超級市場之後，在經過超級市場 Reader 的讀取之後，可以很快得知該貨品是什麼，並且送到陳列該類別商品的地方，而超市也能夠精確地掌

握每一項貨品的存貨量、送達日期、擺放位置以及距離保存期限日期等資訊。

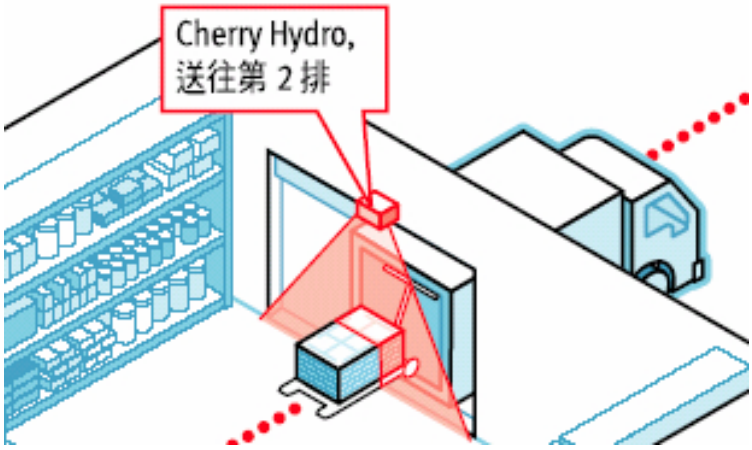

圖 10　將每一個貨品送達到其擺放的架上[2]

6. 顧客在購買完商品之後，經過出口時，放置在出口的 Reader 會將顧客買的商品一一讀入，計算顧客所需支付的總價，不需要排隊結帳，有效地加快顧客的購物流程。

圖 11　讀取標籤並計算總消費額[2]

7. 當顧客從架上拿走商品時，因為在架上也有裝置 Reader，所以架上剩有多少商品，超級市場市的存貨管理系統便能及時地了解每一項商品的存貨量為多少，如果一但存貨量低於標準值，就能夠很快地補貨，如此也同時能降低倉儲的成本。

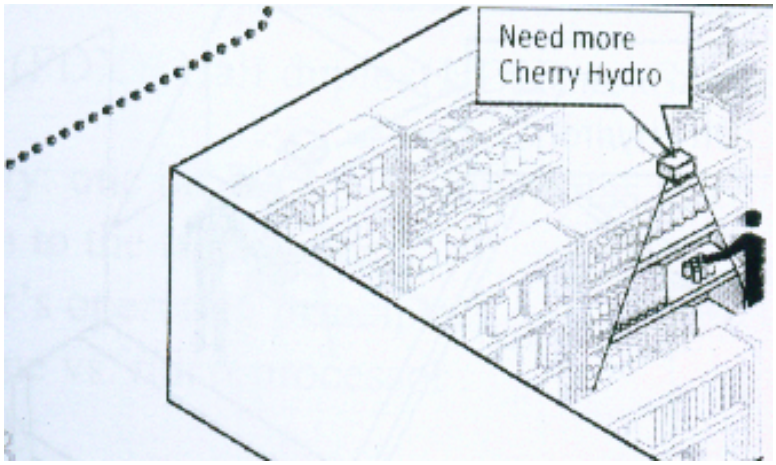

圖 12　重新計算架上的庫存數[2]

2.2　辨識演算法

RFID Reader 和 Tag 之間的資料傳遞，並非使用我們所熟悉的 TCP/IP 網路傳輸模式，而是使用無線電波來傳遞資料，而且資料的傳遞是雙向的全雙工模式，也就是 Reader 和 Tag 可以同時傳遞和接收資料，而傳統的 Bar Code 則是單向的傳輸模式。

有關於 RFID Reader 和 Tag 之間的辨識方式和演算法，EPC Global 提出了兩個基本的辨識演算法，即為 Binary Tree 辨識演算法[6,7]和 Query Tree 辨識演算法[6,7]，而現行廠商所自行製造的 RFID Reader 和 Tag，其所使用的辨識演法都是以 EPC Global 所提出的這兩個演算法為基礎而發展的。

而 EPC Global 所提出的這兩個辨識演算法在許多方面有著共同點，這些共同點也漸漸形成 RFID 辨識演算法發展的基礎和共識，而這些共同點包含了下列各點：

- Reader 和 Tag 之間的溝通採一問一答的方式。
- Reader 根據 Tag 的 Response 來決定接下來的詢問。
- Reader 和 Tag 之間的問答必須持續到所有的 Tag 都被辨識出來為止。

■ Tag 的 Response 一次僅傳送一定的 Bit 數 。

至於 Tag 的 ID Code，我們可以用一個 N 階層的 Full Binary Tree 來表示，如下圖所示，這是一張 3 Bits Tag 的 ID Code 表示圖，最左邊的 ID Code 即為 000，最右邊的 ID Code 為 111，其餘可類推之。

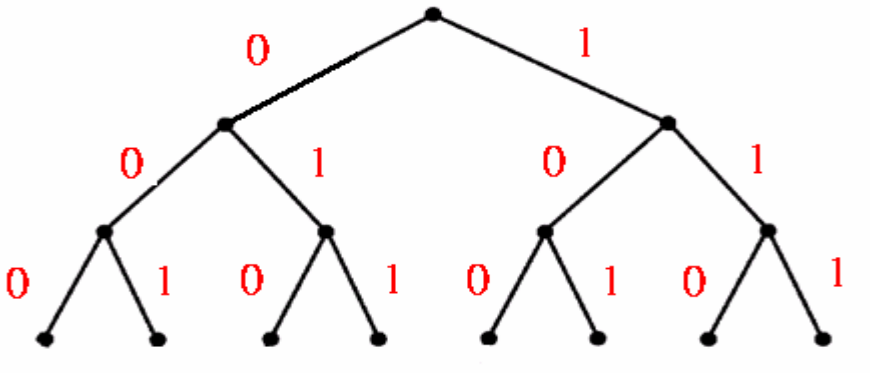

圖 13 ID Code Tree

2.2.1 Binary-Tree 演算法

在 Binary Tree 演算法中[6,7]，每次的查詢，Reader 會廣播 0 或 1 的 Bit 給所有的標籤，而查詢的順序則是從標籤最左邊的 Bit 開始查詢起，當標籤收到 Reader 廣播所送來的 Bit 後，會將這個 Bit 的值(0 或 1)與標籤本身 ID 所對應的 Bit 值相比對，當然，Reader 必須告知標籤現在所查詢的是哪一個位置的 Bit，比對之後，如果兩者的值相同，那麼標籤便會回送 Tag 自身 ID 的下一個 Bit 值給 Reader，如果兩者的值不相同，該標籤就會進入靜止狀態，且不再做任何回應的動作，直到下一輪的查詢為止。

舉例來說假設一開始第一次 Reader 先 Broadcast 第一個 Bit，其值為 0，當每個 Tag 收到這一個從 Reader 廣播傳送過來的 Bit，Tag 要先知道這是在詢問第幾個 Bit，然後再跟自己的 ID Code 中相對應的 Bit 做比對，如果比對的結果不同，該 Tag 就會進入靜止狀態，直到下一輪查詢才回復成 Active 模式，如果比對的結果相同，Tag 就會將自身 ID Code 的下一個 Bit 的值傳送給 Reader。

由於 Reader 和 Tag 在做資料傳輸時，Bit 0 和 Bit 1 所使用的 Channel 並不相同，所以傳送 Bit 0 的 Tag 和傳送 Bit 1 的 Tag 並不會發生碰撞，但如果同時有多個 Tag 傳送 Bit 0 或 Bit 1 給 Reader，那麼就會發生碰撞，而此時 Reader 也就同時得知有兩個以上的 Tag 符合目前的查詢。

接著 Reader 會再繼續發出下一個查詢 Bit，並一直重複上述的步驟，直到辨識出一個 Tag 為止即完成一輪的辨識工作，所以在 Binary Tree Algorithm 中，每一個 Round 結束都可以辨識出一個 Tag，如此一直重複相同的步驟，直到辨識出所有的 Tag 為止。

EPC Global 之所以會把 Binary Tree 辨識演算法當作是 RFID Reader 和 Tag 辨識溝通過程的基礎，主要是因為 Binary Tree 辨識演算法有著以下幾項的優點，包括：

- Tag 演算法簡易，不需複雜的電路設計
- Tag 不需要有記憶體，符合 Tag 便宜大量化的原則
- 此辨識演算法不受 Tag 長度的影響

實例解說

根據前段所描述 Binary Tree 演算法的原理和規則，接下來我們用一個實際運作的例子來說明整個 Binary Tree 演算法的辨識流程。

假設現在一共有四個 Tag，Tag A、Tag B、Tag C 和 Tag D 同時要給 Reader 來做辨識，而這四個 Tag 的 ID Code 分別為：

Tag A： 0 0 1
Tag B： 0 1 1
Tag C： 1 0 0
Tag D： 0 1 0

而整個的辨識流程將如下表所示：

表 2　Binary Tree 演算法辨識步驟

Reader 送出	Tag 回應	被辨識出的 ID
0	*	
0	1	001
0	*	
1	0 , 1	010
0	1	
1	1	011
0	-	
1	0	
0	0	100

步驟一：首先 Reader 先 Broadcast 第一個 Bit，值為 0，此時 Tag A、Tag B 和 Tag D 的第一個 Bit 都為 0，所以也都傳送他們的下一個 Bit 的值給 Reader，所以在 Channel 1 便發生碰撞，而 Tag C 因為第一個 Bit 不是 0，所以進入靜止狀態。

步驟二：接下來 Reader 又 Broadcast 第二個 Bit，其值為 0，此時只有 Tag A 的第二個 Bit 是 0，所以 Tag A 傳送他的下一個 Bit 給 Reader 也就是 1，Reader 收到後便順利地辨識出第一個 Tag A。

步驟三：接著 Reader 繼續 Broadcast 第一個 Bit，值為 0，因為 Tag B 和 Tag D 的第一個 Bit 都為 0，所以也都傳送他們的下一個 Bit 的值給 Reader，所以在 Channel 1 便再次發生碰撞，而 Tag C 因為第一個 Bit 不是 0，所以再次進入靜止狀態。

步驟四：接下來 Reader 又 Broadcast 第二個 Bit，其值為 1，Tag D 會回應 0，此時 Reader 就辨識出 Tag D，雖然 Tag B 也同時回應了 1 給 Reader，但是因為 Reader 一個 Round 只辨識一個 Tag，所以 Tag B 在此次查詢並未被辨識出來。

步驟五：接下來第三個 Round 開始，Reader 繼續 Broadcast 第一個 Bit，其值為 0，而此時只剩下 Tag B 有回應 (Tag A 和 Tag D 已被辨識)，其回應的值為 1，而 Tag C 因為第一個 Bit 不是 0，所以再次進入靜止狀態。

步驟六：Reader 繼續 Broadcast 第二個 Bit，值為 1，這時仍就只有 Tag B 有回應，回應的值為 1，而 Reader 也同時在這次查詢後，辨識出 Tag B，也就是 ID Code 為 0 1 1。

步驟七：Reader 廣播第一個 Bit，其值為 0，因為 Tag A、Tag B 和 Tag D 都已經被辨識，所以這次 Reader 的查詢便沒有任何 Tag 有回應 (Tag C 的第一個 Bit 為 1，所以不會回應)。

步驟八：最後第四個 Round 開始，Reader 廣播第一個 Bit，其值為 1，這時 Tag C 經比對第一個 Bit 之後，因為相符所以 Tag C 會回應下一個 Bit，其值為 0。

步驟九：最後，Reader 廣播第二個 Bit，其值為 0，這時 Tag C 經比對之後仍然符合，所以 Tag C 回應自身 ID Code 的下一個 Bit 0，而在這一次查詢後，Reader 就辨識出最後一個 Tag C，所有的 Tag 辨識工作便告完成。

這個部份主要是介紹 Binary Tree 演算法如何在 RFID 中用來讓 Reader 辨識每一個 Tag，並且透過這一個簡單的例子來詳細說明每一個步驟的過程是如何進行的。

2.2.2 Query-Tree 演算法

EPC Global 除了提出先前所介紹的 Binary Tree 辨識演算法之外，另外還提出了 Query Tree 辨識演算法[6,7]，Query Tree 辨識演算法的邏輯和 Binary Tree 辨識演算法的差異並不大，Query Tree 辨識演算法甚至是由 Binary Tree 演算法演變而來的，但卻在一些少數的差異上造成 Reader 和

Tag 在設計上的明顯不同。

Query Tree Algorithm 包含了許多回合的查詢與回應，每一個回合，Reader 會 Broadcast 一個 ID 字串，收到該 ID 字串後，Tag 會檢查自己的 ID Code 是否包含該 ID Code，如果有的話，Tag 就會回應剩餘的 ID Code 給 Reader，若有超過一個以上的 Tag 回應給 Reader，那麼就會有碰撞發生，而 Reader 也就一並得知某 ID 字串開頭的 Tag 不只一組，接著 Reader 會在先前 Broadcast 的字串中 Append 新的 0 或 1 的 Bit，然後再 Broadcast 出去，直到只有一個 Tag 回應為止，而完成一個 Round 的查詢，並辨識出唯一的一個標籤。接下來來 Reader 便會一值重複這樣的步驟來進行查詢的動作，直到所有的 Tag 都被辨識出來為止。

舉例來說，一開始 Reader 廣播第一個 ID Code 字串 0，這時當 Tag 收到此 ID Code 字串後，就和自己的 ID Code 開頭比對，如果比對後相同，就回應通知 Reader，如果超過一個 Tag 有回應，那麼就會發生碰撞，而此時 Tag 也同時得知以該 ID Code 開頭的 Tag 在兩個以上。

接著，Reader 會在先前的 ID Code 字串加上第二個 Bit 變為 00，同時再廣播出去，Tag 收到後，再繼續進行比對的動作，如果又和自身的 ID Code 的起始 Bits 吻合，那麼就會再次回應通知 Reader，然後 Reader 就再加入一個 Bit 至先前的查詢 ID Code 字串，直到 Reader 所廣播的查詢 ID Code 字串只有一個 Tag 有回應，而結束一個 Round 的查詢，並辨識出一個 Tag。Reader 持續重複相同的動作，直到所有的 Tag 都被辨識出為止。

而 Query Tree 演算法和 Binary Tree 演算法最大的不同點包括以下的幾點：

- Reader 的查詢字串會逐漸變長
- Tag 為了要能比對超過一個 Bit 的值，所以 Tag 需要有較多的記憶體
- Tag 回應是否包含 Reader 所廣播的查詢字串，且回應自身 ID Code 所有剩餘 Bit 的值

實例解說

根據前段所描述 Query Tree 演算法的原理和規則，接下來我們用一個實際運作的例子來說明整個 Query Tree 演算法的辨識流程。

假設現在一共有四個 Tag，Tag A、Tag B、Tag C 和 Tag D 同時要給 Reader 來做辨識，而這四個 Tag 的 ID Code 分別為：

Tag A： 0 0 0
Tag B： 0 0 1
Tag C： 1 0 1
Tag D： 1 1 0

而整個的辨識流程將如下表所示：

表 3　Query Tree 演算法辨識步驟(*表發生碰撞)

Reader 送出	Tag 回應	被辨識出的 ID
0	*	
00	*	
000	A	000
0	01	001
0	-	
1	*	
10	1	101
1	10	110

步驟一：首先 Reader Broadcast 第一個查詢 ID 字串 0，此時因為 Tag A 和 Tag B 的 ID Code 開頭均符合查詢字串，所以這兩個 Tag 都會回應，並發生碰撞。

步驟二：接著，Reader 在原本的查詢 ID 字串加入一個 Bit 0 而成為 00 的查詢字串，而 Tag A 和 Tag B 的 ID Code 開頭，又再次符合查詢字串，所以這兩個 Tag 會再次回應，並再次發生碰撞。

步驟三：Reader 在原本的查詢 ID 字串再加入一個 Bit 0 而成為 000 的查詢字串，而此時只有 Tag A 符合這樣的 ID Code，所以只有 Tag A 有回應，Reader 也在這一個步驟中辨識出 Tag A。

步驟四：第二個 Round 的查詢開始，Reader Broadcast 第一個查詢 ID 字串 0，而現在只剩下 Tag B 的 ID Code 起始值符合查詢查字串，Tag B 便回應 Reader 剩餘的 ID Code 01，而 Reader 也在這次的查詢辨識出 Tag B。

步驟五：Reader 廣播第一個 ID 查詢字串 0，因為 Tag A 和 Tag B 都已經被辨識，所以這次 Reader 的查詢便沒有任何 Tag 有回應，Reader 也確定已無任何 Tag 是以 ID Code 0 為起始值。

步驟六：在第三個回合的查詢中，Reader 首先廣播第一個 ID 查詢字串 1，此時因為 Tag C 和 Tag D 的 ID Code 開頭均符合查詢字串，所以這兩個 Tag 也都會回應並且發生碰撞。

步驟七：接下來，Reader 在原本的查詢 ID 字串加入一個 Bit 0 而成為 10 的查詢字串並且廣播出去，而此時只有 Tag C 的 ID Code 起始值符合 Reader 所送出的 ID 查詢字串，所以在這一次的查詢只有 Tag C 有回應，回應值為 1，而 Reader 也在這次的查詢中辨識出 Tag C。

步驟八：第四個回合的查詢開始，Reader 首先廣播第一個 ID 查詢字串 1，而現在只剩下 Tag D 的 ID Code 起始值符合 Reader 所送出的 ID 查詢字串 (其餘都已被辨識完成)，所以 Tag D 就回應剩餘的的 ID Code 給 Reader，也就是 10，而 Reader 也在這次的查詢中辨識出 Tag D，並完成所有 Tag 的辨識工作。

2.3 討論

經過以上的實例解說，我們能夠很清楚地了解到 Query Tree 演算法到底是如何運用在 RFID 的 Reader 和 Tag 之間的辨識流程。而 EPC Global 所提出的這兩個辨識演算法目前已經成為 Reader 和 Tag 之間辨識的基礎演算法，對製造 RFID 標籤和 Reader 的廠商來說，也是以其中一個演算法為基礎去修改成適合它們產品的演算法。

在 Query Tree Algorithm 的演算法中，因為每一個 Tag 都要能比對超過一個 Bit 的 Id Code 值，所以在 Tag 的設計上，必須要有一些 Memory 來儲存當下的資訊，所以如果運用 Query Tree Algorithm 作為 RFID 的辨識演算法，可能 Tag 的記憶體成本會相對提高一些。而 Binary Tree Algorithm 則不需要讓 Tag 記憶任何 Reader 的查詢資訊，所以相對來說，如果用 Binary Tree Algorithm 當作 RFID 的辨識演算法，Tag 的成本則可以下降，也較為符合 RFID 便宜且大量化的基本概念。

另外，以辨識的速度來說，Query Tree Algorithm 會有較好的表現，也就是說 Query Tree Algorithm 辨識的速度會比較快，而 Binary Tree Algorithm 的辨識速度會比較慢，原因是因為在 Query Tree Algorithm 中，Reader 每次查詢的 Bit 數比較多，所以速度就來的比較快。所以如果在整個辨識過程中，有一定的時間限制，或是所要同時辨識的 Tag 數量相當可觀，那麼使用 Query Tree Algorithm 作為 RFID 的辨識演算法，或許會是比較好的選擇。

而本篇論文所提出的辨識機制是選擇以 Binary Tree 辨識演算法為基礎，而未選用 Query Tree 辨識演算法，主要有下列幾點原因：

- Query Tree 辨識演算法的記憶體成本落在 Tag 身上。
- Query Tree 辨識演算法的查詢和回應都非單一位元，資訊不易掌握。
- 已標準化的 Auto-ID，是選用 Binary Tree 辨識演算法。

第三章　新式標籤辨識機制

在這一個章節裡，本論文將會提出一個新的辨識機制，而這個機制是特別針對 Binary Tree 辨識模式會面臨到的一些問題所作的解決方案，在本章節裡，除了描述新的辨識機制的邏輯之外，也會在最後以一個例子來解說本章節所提出的辨識機制的運作流程。

3.1　問題分析

在非單一標籤的辨識環境下，在 Reader 辨識標籤的過程裡，Reader 和標籤的總溝通次數決定了整個辨識過程所需的時間以及所需要消耗的總電力，如果能夠將 Reader 和標籤的總溝通次數降低，對於整體所需的辨識時間和所消耗的電力一定能夠減少。

以 Binary Tree 辨識演算法為例，其實在整個辨識過程中有一些步驟是可以預期並且加以簡化的，我們用面的幾個例子來說明：

情形一：假設現在一共有三個標籤，Tag A、Tag B 和 Tag C 同時要給 Reader 來做辨識，而這三個 Tag 的 ID Code 分別為：

Tag A：　0 0 0

Tag B：　0 1 0

Tag C：　1 1 1

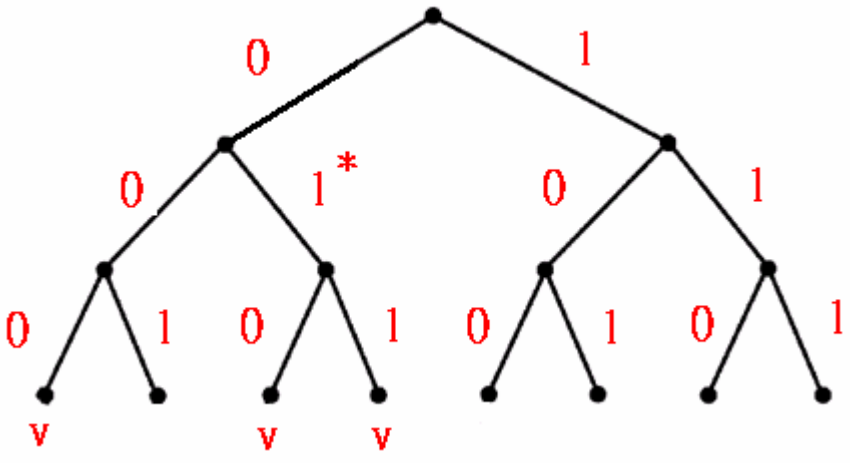

圖 14　辨識狀況一 (*表碰撞發生)

當 Reader 開始第一次查詢時，廣播的查詢位元為 0，而這時 Tag A、Tag B 和 Tag C 在收到 Reader 的查詢位元之後，經比對自身 ID Code 的第一個位元值後，因為三個標籤均符合，所以全都回應自身 ID Code 的下一個位元值給 Reader，Tag A 回應 0，Tag B 回應 1，Tag C 也回應 1。

因為回應 0 和 1 所使用的 Channel 不同，所以僅在 Channel 1 發生碰撞。接下來 Reader 繼續第二個位元值的查詢，直到辨識出第一個標籤(Tag A)為止，並且完成第一個回合的查詢。

當第二回合的查詢開始，Reader 又再次以查詢位元 0 作為第一次的查詢值，而 Tag B 和 Tag C 當然也再次符合這次的查詢並且向 Reader 回應，Tag B 回應 1，Tag C 也回應 1，所以在 Channel 1 又再次的發生碰撞。

仔細注意這一次查詢動作的結果，我們不難發現，第二回合的第一次查詢結果其實和第一回合的第一次查詢結果在 Channel 1 的回應狀況是完全相同的，也就是說如果 Reader 能夠將先前的查詢狀況納入參考，那麼以這個例子來說，第二回合的第一次查詢其實是可以省去的，也就是可以直接從第二個位元開始查詢起，而省去一次的查詢動作。

情形二：假設現在一共有三個標籤，Tag A、Tag B 和 Tag C 同時要給 Reader 來做辨識，而這三個 Tag 的 ID Code 分別為：

Tag A： 0 0 0
Tag B： 1 0 0
Tag C： 1 1 0

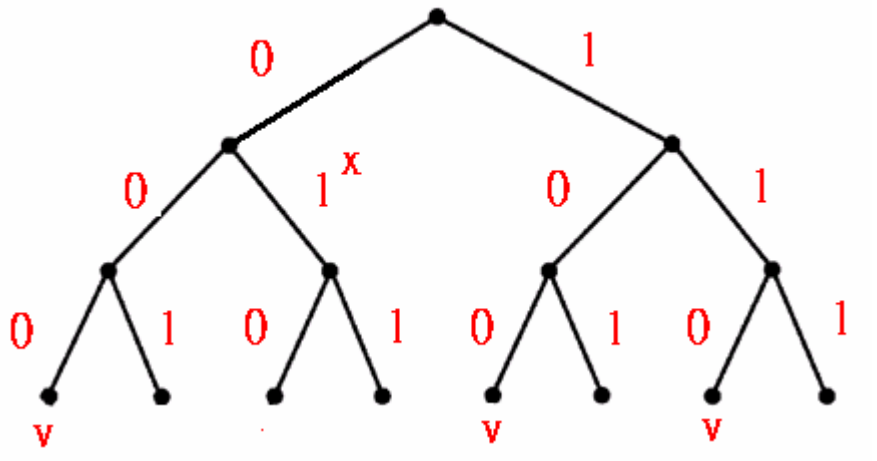

圖 15　辨識狀況二(x 表無回應)

當 Reader 開始第一次查詢時，廣播的查詢位元為 0，而這時 Tag A、Tag B 和 Tag C 在收到 Reader 的查詢位元之後，經比對自身 ID Code 的第一個位元值後，因為只有 Tag A 的第一個 Bit 為 0，所以只有 Tag A 會回應自身 ID Code 的下一個位元值 0 給 Reader，Tag B 和 Tag C 都不會回應。

因為沒有任何標籤在 Channel 1 上做回應，所以僅在 Channel 0 會收到回應訊息。接下來 Reader 繼續第二個位元值的查詢，直到辨識出第一個標籤(Tag A)為止，並且完成第一個回合的查詢。

當第二回合的查詢開始時，Reader 又再次以查詢位元 0 作為第一次的查詢值，但是現在只剩下 Tag B 和 Tag C 尚未被辨識，但是因為 Tag B 和 Tag C 的第一個 Bit 都是 1，所以在這一次的 Reader 查詢，兩個 Tag 都不會做回應。而 Reader 在這一次的查詢因為沒有任何的標籤有回應，所以在下一個回合的查詢，Reader 就會改以位元值 1 作為接下來的查詢字元，開始查詢剩餘尚未被辨識的標籤。

在 Reader 改以位元值 1 作為該回合第一次的查詢值之前，Reader 最後一次的查詢結果沒有任何標籤有回應，而這一次的查詢結果其實在第一回合的第一次查詢就表現在 Channel 1 的回應結果上，因為沒有任何標籤在 Channel 1 上有回應，所以在 Reader 辨識出 Tag A 之後，就可以確定沒有任何標籤的 ID

Code 是以 0 起始的，以這個例子來說，第二回合的第一次查詢其實是可以省去的，也就是可以直接從位元值 1 作為第一個位元的查詢值，而省去一次的查詢動作。

以 EPC Global 所提出的 Binary Tree 演算法辨識架構而言，Reader 辨識各個標籤的過程有著以下兩點特色：

- Reader 每一回合只辨識一個標籤
- Reader 辨識一個標籤需要查詢 n-1 次，n 為標籤的 Bits 長度

所以 Reader 要完成 N 個標籤的辨識工作，一共需要 N(n-1)次的查詢動作，而標籤的總回應次數，則和標籤的 ID Code 值有關，如果標籤的前幾個 ID Code 值相似度高的話，標籤的總回應次數就會比較多，反之，標籤的總回應次數就會比較少。

3.2 發展的機制

3.2.1 回溯演算法(Back Tracking Algorithm)簡介

回溯演算法(Back Tracking Algorithm)[15]是一種把先前所發生的經驗作為依據，來判斷是否要進行之後剩餘的嘗試，如果先前的經驗或結果已經可以告訴我們繼續下去也不可能成功的話，那麼就放棄目前正在嘗試的方法，而選擇另一種方式來進行，省去一些無謂的時間和努力；反之，如果先前的經驗或結果無法證明此方法不可能成功，那麼就選擇繼續嘗試下去，直到成功或發現不可能成功為止。

回溯演算法最典型的例子就是所謂的八后問題，問題本身的描述是：我們要在一個 8 X 8 的棋盤上，擺上八個皇后棋子，並且規定每一個皇后棋子在棋盤上的行或者是列都不能和其他的皇后棋子一樣。在這個問題中，因為棋盤是 8 X 8，那麼每一行可以有 8 種擺法，一個棋盤又有 8 列，所以所有可能的擺法一共有：

8x8x8x8x8x8x8x8
=8^8
=(2^3)^8
=2^24
=１６７７７２１６種擺法

再思考一下，如果第一個棋子已經擺在第一行了，那麼以後就不可以再有棋子擺在第一行了，也就是以先前棋子的擺放方式作為之後擺放棋子的依據，如此下去，我們就可以估算用此法最多需要：

8*7*6*5*4*3*2*1
=8!
=４０３２０種擺法

使用回溯演算法的結果和原始的方法差了約４１６倍，也就是說第一個方法的時間，可以讓第二個方法算４１６次。

3.2.2 機制說明

在此，本論文將提出一個以 Binary Tree 演算法為基礎的新式辨識機制，而這新式辨識架構是把回溯演算法(Back Tracking Algorithm)的概念和 Binary Tree 辨識演算法結合在一起，而達到減少 Reader 查詢次數和標籤回應次數，進而達成縮短 Reader 辨識標籤所需的時間和節省所需的電力。

新式辨識機制最主要的觀念在於將 Reader 查詢結果的資訊保留起來，以作為之後 Reader 查詢動作的參考依據，這對於 Reader 要選擇 0 或 1 的查詢位元值以及要查詢標籤的第幾個位元將有所幫助。

因為標籤回應 Reader 的頻道有兩個，一個用來傳送位元值 0，一個用來傳送位元值 1，而在每一次 Reader 的查詢動作發生後，這兩個頻道都同時會有標籤回應的結果，而且結果只有三種可能，包括：

- 碰撞 (超過兩個標籤在該頻道有回應)
- 只有一個標籤回應
- 沒有任何標籤回應

根據 Reader 和標籤在訊息交換上的特性，如果我們能把每回合的查詢在這兩個頻道上的結果適時的儲存起來，那麼 Reader 在不同回合的查詢中，如果需要做和之前相同的查詢的話，這個查詢步驟就可以省去。

3.2.3 方法程序

而本論文所提出的新式標籤辨識機制的程序和步驟如下說明：

步驟一：在每一次 Reader 的查詢過程中，Reader 必須保留每個標籤位元的不同查詢位元值的查詢結果在兩個頻道上的回應狀況，例如標籤第一個位元查詢值為 0 在兩個頻道上的標籤回應結果。

步驟二：Reader 只需要記住在頻道上的兩種回應狀況，一種為在頻道上發生碰撞，另一種是在頻道上沒有任何的標籤有回應。如果 Reader 在之後又再次要查詢相同的標籤位元和位元值，Reader 就會檢查之前的查詢結果。

步驟三：如果之前的查詢結果是發生碰撞的，Reader 就可以跳過該位元的查詢，直接向下一個位元作查詢。

步驟四：如果先前的查詢結果是沒有任何標籤有回應的狀況，Reader 便可以跳過以該位元值為起始值的所有位元查詢。

步驟五：倘若先前沒有任何的查詢紀錄，Reader 就要將該次的查詢結果儲存下來，以供之後的查詢參考之用，並且重複這些相同的步驟直到辨識完所有的標籤為止。

3.2.4 新式辨識機制流程圖

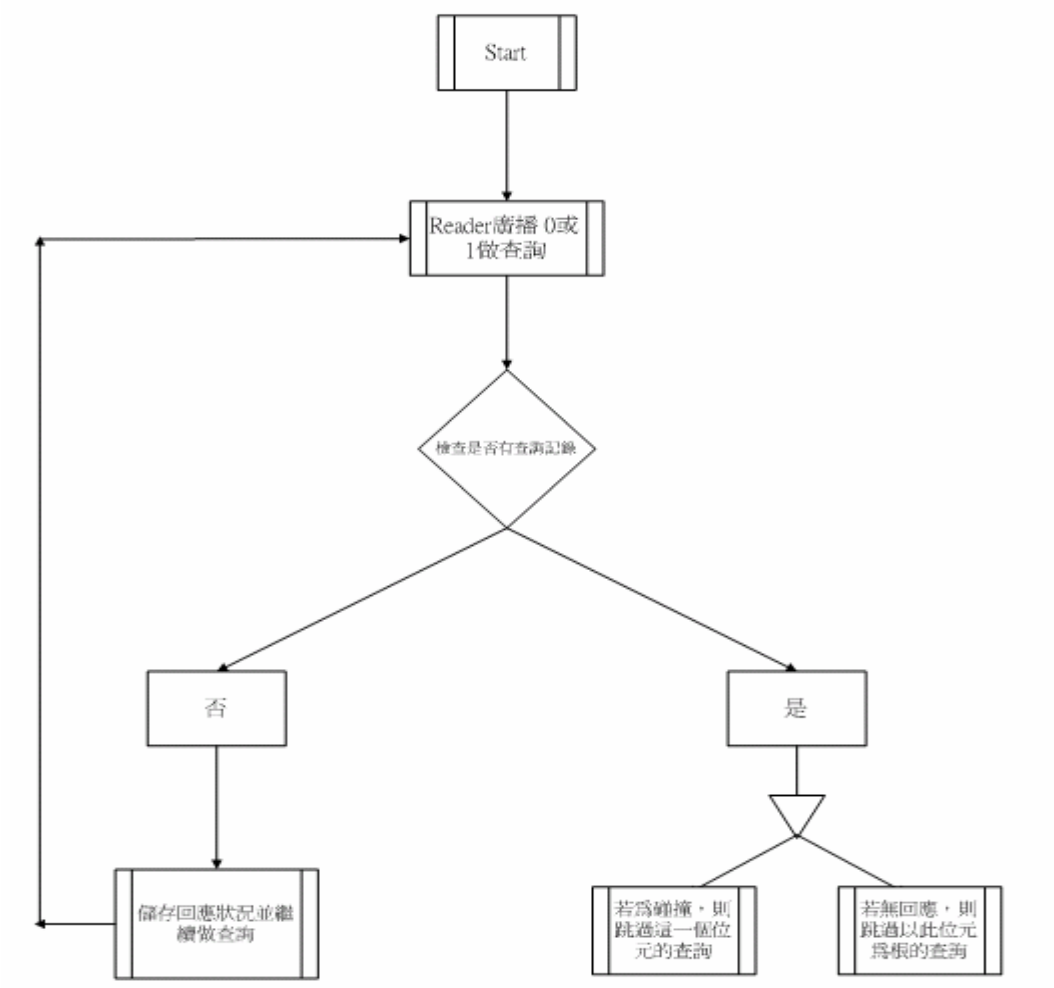

圖 16 新式辨識機制流程圖

3.2.5 實例解說

根據前段所描述的新式辨識機制，現在我們用一個實例來說明此辨識機制和原始的 Binary Tree 演算法的辨識流程有何不同。

假設現在一共有四個 Tag，Tag A、Tag B、Tag C 和 Tag D 同時要給 Reader 來做辨識，而這四個 Tag 的 ID Code 分別為：

Tag A： 0 0 0
Tag B： 0 0 1
Tag C： 1 0 0
Tag D： 1 0 1

而整個的辨識流程將如下表所示：

表 4　新式辨識機制的辨識步驟

Reader 送出	Tag 回應	被辨識出的 ID
0	A , B	
0	A , B	000
0	B	001
1	C , D	
0	C , D	100
0	D	101

步驟一：首先 Reader 先 Broadcast 第一個查詢位元，值為 0，此時 Tag A 和 Tag B 的第一個 Bit 都為 0，所以也都傳送他們的下一個 Bit 的值給 Reader，所以在 Channel 0 便發生碰撞。

步驟二：接下來 Reader 又 Broadcast 第二個查詢位元，其值為 0，此時 Tag A 和 Tag B 的第二個 Bit 仍為 0，所以也都傳送他們的下一個 Bit 的值給 Reader，Tag A 傳 0，Tag B 傳 1，Reader 也在此次查詢辨識出第一個 Tag A。

步驟三：根據 Reader 第一次的查詢經驗，所以第二個回合，Reader 就會從第二個位元開始查詢起，查詢的位元值為 0，而此時只有 Tag B 符合，回傳值為 1，所以 Reader 在這次查詢辨識出 Tag B。

步驟四：因為 Reader 第一次查詢時，在 Channel 1 上沒有任何標籤有回應，所以當 Reader 辨識出 Tag B 之後，在這一回合的查詢就不再查詢位元值 0 了，而改為查詢位元值 1(第一個位元)，而這時 Tag C 和 Tag D 同時符合，所以也都傳送他們的下一個 Bit 的值給 Reader，所以在 Channel 0 又再次發生碰撞。。

步驟五：接下來 Reader 又 Broadcast 第二個查詢位元，其值為 0，此時 Tag C 和 Tag D 的第二個位元仍為 0，所以也都傳送他們的下一個 Bit 的值給 Reader，Tag C 傳 0，Tag D 傳 1，Reader 也在此次查詢辨識出 Tag C。

步驟六：根據 Reader 剛才的查詢經驗，所以這個回合，Reader 就會從第二個位元開始查詢起，查詢的位元值為 0，而此時只剩下 Tag D 合，回傳值為 1，所以 Reader 在這次查詢辨識出 Tag D，並完成所有標籤的辨識。

以這個例子來說，新式辨識機制大約比原本的 Binary Tree 辨識模式少了兩個步驟左右，如果把標籤的位元數增加，數量便多，相信能減少一定比例的查詢次數。

第四章　模擬結果與分析

在本章節裡，本論文將會先模擬實作原本 Binary Tree 辨識演算法，然後再模擬本論文在上個章節所提出的新辨識機制，之後將討論系統完成後所做的模擬實驗結果以及針對模擬結果所做的分析，並且在最後根據所產生的模擬結果做出系統的評估與建議。

4.1　測試平台與環境

本論文之實驗環境架構於實驗室的軟硬體環境，並且由三台不同軟硬體環境的電腦分別測試之，以降低實驗環境所造成的誤差，其硬體設備與軟體環境列出如下：

硬體設備：

CPU：　AMD Athlon　2.5G　，　AMD Athlon　1.3G
Intel Pentium4 2.4G　，　Centrino 1.3G

記憶體：　256MB　，　384MB　，　512MB　，　256MB

硬碟大小：　30G Bytes

網路介面卡：　3COM 905C-TX　，　D-Link 530TX
Realtek 8139　，　Intel PRO VE

作業系統：

Windows 2000 Pro　，　Windows XP
Windows 2003 Server　，　BSD 5.1 Release

網路環境：

交通大學校園網路：　乙太網路　10MBps / 100MBps
無線網路：　802.11b

開發平台：

Windows XP
JDK 1.4.1
J-Builder 8.0

程式語言：

Java Language

4.2 模擬假設與限制

4.2.1 模擬假設

在本章節所做的模擬實驗中，有做一些基本的假設條件，而模擬的結果也是以這些假設條件為基礎，這些基本假設茲列如下：

1. 假設標籤的位元長度為有限：

標籤的位元長度是影響辨識過程中，Reader 和 Tags 的總溝通次數的主因之一，而總溝通次數也直接影響了辨識過程所需的總時間，位元長度愈長，所需的辨識時間愈多，兩者為正比的關係，而本模擬實驗最多只模擬到 96 個 Bits 的標籤長度，和 EPC Global 所制定的標籤位元長度標準相同。

2. Reader 和標籤之間的溝通媒介穩定：

本實驗的另一個假設是假定 Reader 和所有標籤之間傳輸媒介是穩定的，因為 Reader 和標籤的傳輸媒介是空氣，所以這個假設主要是把溫度、輻射干擾、空氣擾動(例如：風)和溼度等天然不可抗拒的因素排除在外，也因為這些因素所造成的影響極為細微，所以忽略這些天然因素，所造成的模擬誤差會非常有限。

3. Reader 和標籤均處於靜止狀態：

另外，本模擬實驗也假設 Reader 和所有的標籤都是處於靜止的狀態下，因為 Reader 和標籤的溝通是透過無線電波，是屬於無線傳輸方式的一種，所以如果 Reader 或標籤其中之一是處於移動狀態的話，就會產生移動前和移動後兩種波形的干擾，造成辨識上的困難，而移動中的標籤辨識並不在本論文的討論範圍內。

4. 所有標籤的位元長度都相同：

除了上述三點之外，本模擬實驗也假設所有要給 Reader 辨識的標籤，其位元長度都要相同，這也是 Binary Tree 辨識演算法的基本假設之一，因為如果標籤的位元長度不完全相同，在辨識的過程中，Reader 無法得知現在所辨識的位元位置是不是標籤的最末位元。而本論文所提出的新式辨識機制是以 Binary Tree 辨識演算法為基礎，所以也就自然承襲了 Binary Tree 辨識演算法的基本假設。

5. 單一個 Reader：

最後，本模擬實驗是假設辨識的過程只有單一個 Reader，所以不會有 Reader 和 Reader 之間的干擾現象；而多 Reader 的辨識模式牽扯到 Reader 和 Reader 之間在做查詢廣播時可能發生的干擾現象，而這個部分的研究不在本論文的討論範圍之內。

本次所做模擬實驗的基本假設如上述五點所列，如果模擬實驗的結果和真實運作的狀況有誤差的話，其發生誤差的主因應該就不外乎上述的五點模擬假設。

4.2.2 實驗限制

除了上述的模擬實驗假設之外，本實驗也同時有一些操作上的限制，茲分述如下：

1. 無法模擬所有標籤的 ID Code 組合：

本模擬實驗中的標籤 ID Code 是以隨機方式產生的，並且保證不會有任兩個標籤的 ID Code 是相同的，由於 ID Code 是隨機產生的，所以無法控制 ID Code 的值，而且，當標籤的位元長度增加時，標籤 ID Code 的所有可能組合幾乎趨近於無限多組，再加上隨機產生的 ID Code 也無偏頗之虞，所以本模擬實驗並沒有考慮將所有可能的 ID Code 都納入實驗。

2. 同時被標籤的辨識數量不超過 512 個：

在 EPC Global 的建議中，當標籤的數量超過 512 個的時候，有可能發生標籤辨識漏失的現象，也就是會有某一些標籤永遠無法被 Reader 辨識出來，故此機制也遵循 EPC Global 的建議，在每次模擬辨識的時候，同時隨機產生的標籤數將會少於 512 個，等同於 Reader 讀取範圍內的標籤數不會超過 512 個的限制。

3. 無法模擬所有可能的標籤位元長度：

在本模擬實驗中，對於標籤位元長度的模擬過程是以漸進式的方式模擬的，也就是將標籤的位元長度漸漸地增加，每一次的增加量為一，但是因為標籤的位元長度不可能無限制地一直增加，所以本論文所做的模擬實驗只進行到 EPC Global 所制定的標準標籤位元長度為止，也就是 96 個 Bit。

4. 有限記憶體：

在本論文所提出的新式辨識機制中，必須要耗費一些記憶體來紀錄前幾次 Reader 的查詢結果，但是當標籤位元長度持續成長時，所需的記憶體也會呈現指數成長的情況，所以在有限記憶體的條件下，紀錄 Reader 每次查詢不同標籤位元位置的數目也會有所限制。

4.3 系統設計

系統各功能模組

在此所做的模擬系統是採用 Java 語言實作出來的，而整個系統主要分成五大功能模組，茲分述如下：

1. 介面模組：

   系統的介面模組包括使用者介面的部份，讓使用者可以自由選擇想要模擬的環境，包括標籤的位元長度、標籤的數量選擇和執行次數等，都可以由使用者來控制；除此之外，系統的介面模組也包括系統模擬結果的輸出，這個部份包含總溝通次數、標籤總回應次數、Reader 總查詢次數、平均溝通次數、標籤平均回應次數、Reader 平均查詢次數以及每一回合被辨識出的標籤的 ID Code。

2. 隨機標籤產生模組：

   在模擬的過程中，系統需要產生一定數量的標籤，而且必須確保每一個標籤的 ID Code 都是不一樣的。從使用者的輸入取得所要產生標籤的位元長度和標籤數之後，系統從 0 到 2 的位元長度次方減 1 的自然數中隨機選取和標籤數等數量的自然數，然後把這些被選取的自然數轉化成二進位的數值，例如自然數 10 將被轉換成 1010。

3. Reader 功能模組：

   Reader 功能模組負責實作出 Binary Tree 辨識演算法以及本篇論文所提出的新式辨識機制的辨識程序，Reader 功能模組要根據這兩個方法的邏輯和程序適時地送出查詢位元以及改變查詢位元的位元值。另外，此功能模組也要根據這兩種辨識架構，在 Reader 要做查詢動作的時候，同時也告知所有標籤該次所要查詢的標籤位置。

4. 標籤功能模組：

標籤功能模組負責實作標籤的回應動作以及適時的操控讓標籤進入靜止模式，並且在每回合開始的時候，讓標籤從靜止模式中 Wake UP 起來，讓辨識工作能反覆進行下去。因為原本的 Binary Tree 辨識演算法以及本篇論文所提出的新式辨識機制，在標籤的演算邏輯上是完全相同的，所以此功能模組不需要時做兩種不同的演算法邏輯。

5. 位元值比對功能模組：

位元值比對功能模組的主要作用是每當 Reader 廣播一個查詢位元後，此功能模組就要把 Reader 所廣播的查詢位元和每一個標籤相對應位元位置的值做比對，如果比對的結果是相同的，就要通知標籤作回應的動作，如果比對的結果是不相同的，那麼就要告訴標籤要進入靜止模式了。此功能模組有兩個重要的參數，一個是比對的位元位置，另一個則是比對的位元值。

4.4 Test Case

本章節所做的模擬實驗一共有四個主要的變因，分別是標籤的位元長度、標籤的總數目、標籤 ID Code 的相似程度，以及新式辨識機制的紀錄位元總數。而本模擬實驗會針對這四個主要的變因進行測試，並且找出 Binary Tree 辨識演算法和新式辨識機制與這四個主要變因之間的關係和其影響程度的大小。

1. 標籤位元長度：

標籤的位元長度直接影響了 Reader 和所有標籤的辨識溝通次數，標籤的位元長度愈長，所需的位元交換次數也就愈多，兩者的關係為正向影響的關係；而本模擬實驗的重點則是要試圖了解當標籤的位元長度逐漸增長時，新式辨識機制所能減少的總溝通次數或比例會隨著標籤位元長度的增減有著什麼樣的變化，而標籤位元長度的增加將以 EPC Global 所制定的標準標籤長度 96 個 Bits 為上限。

2. 標籤總數目：

除了標籤的位元長度之外，所要被辨識的標籤數量也是影響 Reader 和標籤總溝通次數的主要因素之一，標籤的總數量愈多，Reader 和標籤的溝通次數也會隨之增加，兩者的關係仍為正向的影響關係；而在本模擬實驗中，主要是要測試當標籤的總數量逐漸遞增的時候，新式辨識機制所能減少的總溝通次數或比例的相對變化，而測試的總標籤數量將以上述的模擬假設為依據，以 512 個標籤為限。

3. 標籤 ID Code 相似程度：

除了上述的兩項變因之外，標籤 ID Code 的位元值排列情形也是本次模擬實驗所關心的重點之一，所謂的 ID Code 相似程度指的是標籤和標籤之間 ID Code 的漢明距離(Hamming Distance)，在這個變因的實驗中，我們試圖要了解標籤之間的漢明距離長短是否會影響兩種辨識架構中 Reader 和標籤的總溝通次數，以及影響的程度。

4. 紀錄的位元總數：

在先前所提到的模擬限制裡，其中有限的記憶體環境導致無法將所有 Reader 的查詢姐果都予以紀錄下來，所以在這個部分的模擬實驗中，本論文將會以標籤的位元位置為單位逐漸增加，來測試所使用的紀錄位元總數將會對新式辨識機制的改進程度有什麼影響，而每次增加一個標籤位元的紀錄，相對會增加 2 的 n 次方個位元的記憶體(n 為標籤位元位置)。

4.5 模擬結果與數據

4.5.1 針對標籤位元長度變化的模擬結果

標籤總回應次數

下面圖表的模擬結果是以在 300 個標籤的狀況下，改變標籤的位元長度(橫軸)，並且記錄 Binary Tree 辨識架構和本論文所提出的新式辨識機制在各種標籤位元長度下，完成所有標籤的辨識時，標籤的總回應次數(縱軸)，而標籤的位元長度以先前的模擬假設 96 個 Bits 為限。根據下表的統計結果顯示，可以很明顯地發現，新式辨識機制的標籤總回應數要比 Binary Tree 辨識架構的標籤總回應數減少許多。

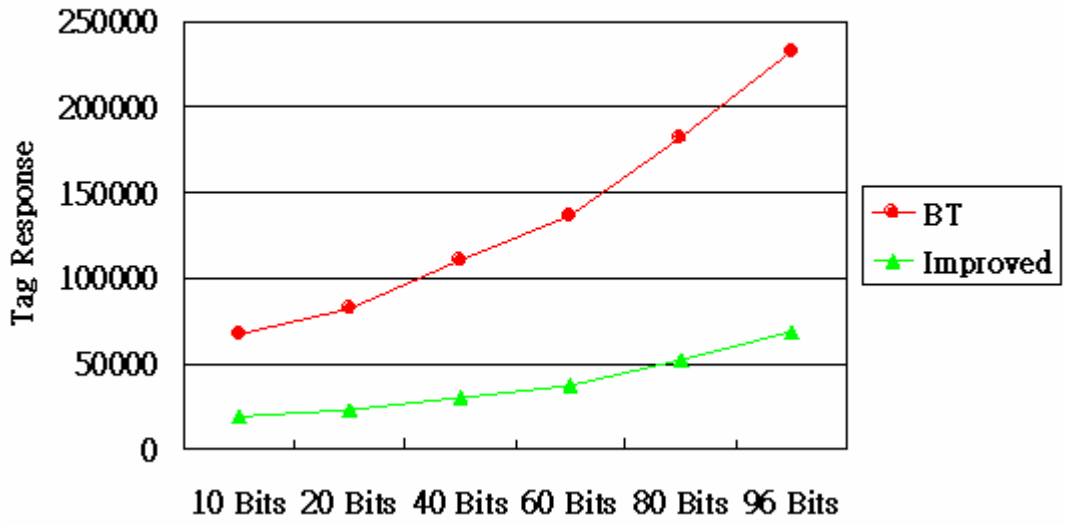

圖 17　針對標籤位元長度改變的模擬結果(一)

假設變數 T1 代表在原本 Binary Tree 辨識架構下，完成標籤辨識過程總共所需的標籤回應次數；並且假設變數 S1 代表在新式辨識機制下，所需的標籤總回應次數，而變數 R1 則代表變數 S1 和變數 T1 的比值(S1/T1)，也就是新式辨識機制所需的標籤總回應次數所占原本 Binary Tree 辨識架構標籤總回應次數的百分比。在上圖 17 中，R1 的值從百分之二十七點八到百分之二十九點五之間不等。

Reader 總查詢次數

除了考慮標籤的總回應次數之外，Reader 在整個辨識過程中對標籤的總查詢次數也是非常重要的辨識效率考量因子，因此，在下面的圖表中，將縱軸的單位改為 Reader 的總查詢次數，檢驗在各種不同的標籤位元長度下，Binary Tree 辨識架構和本論文所提出的新式辨識機制在完成 300 個標籤的辨識後，一共各需要多少次的 Reader 查詢。

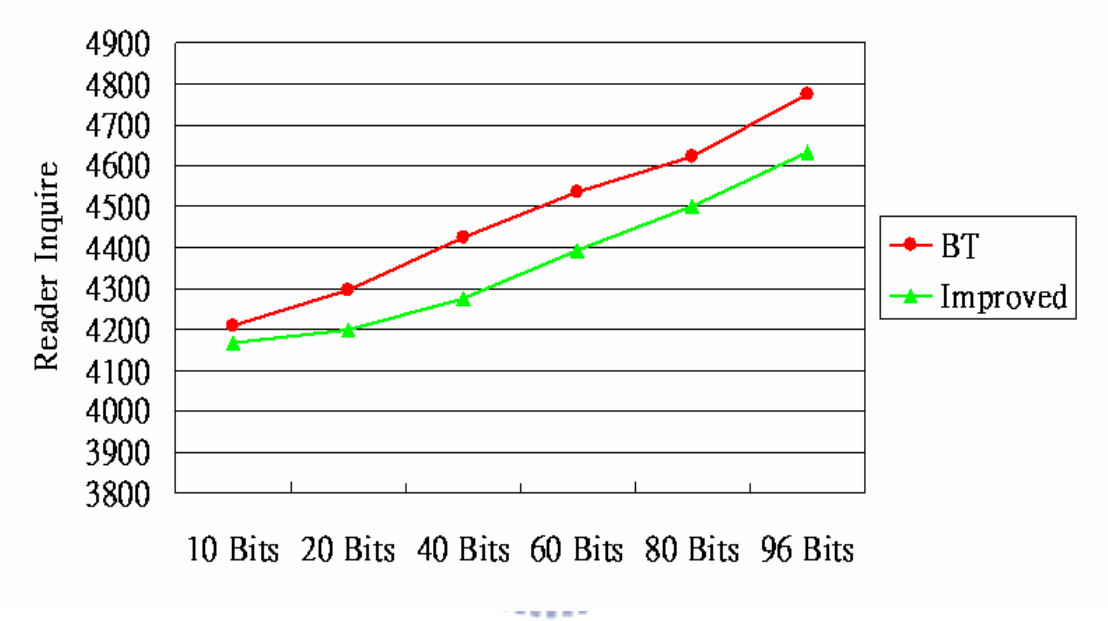

圖 18　針對標籤位元長度改變的模擬結果(二)

假設變數 T2 代表在原本 Binary Tree 辨識架構下，完成標籤辨識過程總共所需的 Reader 查詢次數；並且假設變數 S2 代表在新式辨識機制下，所需的 Reader 總查詢次數，而變數 R2 則代表變數 S2 和變數 T2 的比值(S2/T2)，也就是新式辨識機制所需的 Reader 總查詢次數所占原本 Binary Tree 辨識架構 Reader 總查詢次數的百分比。在上圖 18 中，R2 值的變動範圍在百分之九十八點七到百分之九十六點七左右。

綜合以上的兩個模擬實驗可以發現，本論文所提出的新式辨識機制在標籤的總回應次數上的效能增進幅度比在 Reader 的總查詢次數上有更好的表現，這主要是因為每一次的 Reader 查詢很有可能都伴隨著許多標籤的回應，所以每減

少一次的 Reader 查詢時，同時減少的標籤回應次數很有可能遠超過一次，從這個觀點來解釋標籤回應次數降低的幅度比較大，這樣的實驗結果也就不令人意外了。

4.5.2 針對標籤數量的模擬結果

標籤總回應次數

下面圖表的模擬實驗結果是以標準 96 Bits 的 EPC Code 標籤為辨識對象，逐漸增加標籤的總數量，並且記錄 Binary Tree 辨識架構和本論文所提出的新式辨識機制在這些辨識標籤數量下，完成所有標籤的辨識時，一共需要多少次的標籤回應數。而標籤的總數以先前的模擬假設 500 個 Tag 為上限值。

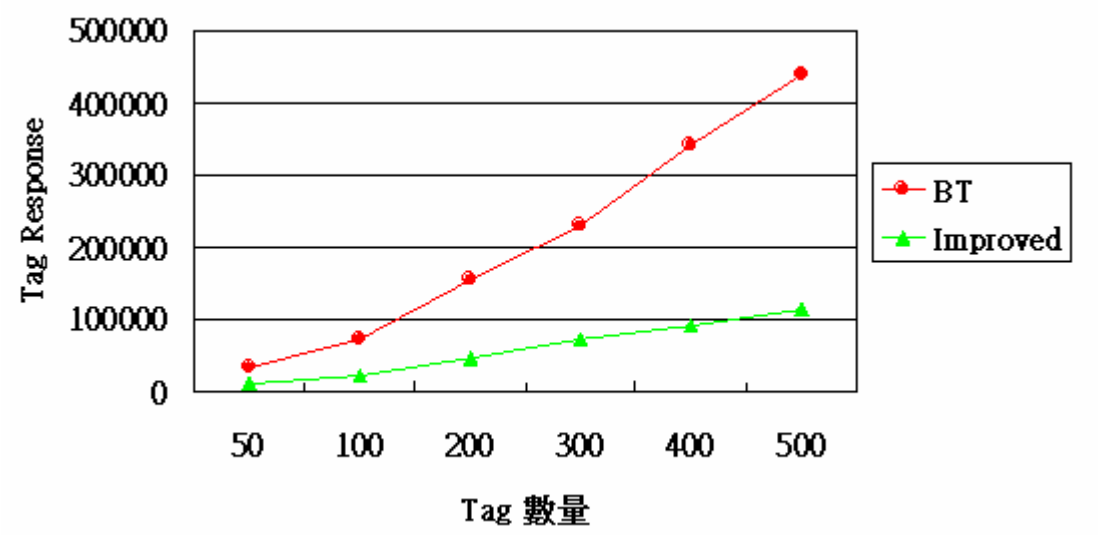

圖 19　針對標籤數量改變的模擬結果(一)

以圖 19 所示的模擬結果而言，在辨識標籤總數量為 50 到 100 個標籤時，R1 的值大約是在百分之三十到三十二之間，而隨著標籤數量持續增加後，在 Binary Tree 辨識架構下標籤總回應次數的增加幅度要比新式辨識機制的增加幅

度大，到標籤總數量為 500 個的時候，R1 的值就大約降到了百分之二十六左右了。

在實際的應用來說，一般的情況，Reader 需要同時辨識的標籤數量，大部分都不會超過 100 個，而要超過 300 個以上的實際應用更是少之又少，而本模擬實驗以 500 個標籤為上限，相信能包含大部份的實際狀況。

Reader 總查詢次數

在這個針對標籤數量多寡的模擬實驗中，也同時把 Reader 的總查詢次數列入觀察的重點，因此，在下面的圖表中，將縱軸的單位改為 Reader 的總查詢次數，檢驗在各種不同的標籤數量下，Binary Tree 辨識架構和本論文所提出的新式辨識機制所需 Reader 總查詢次數的差異。

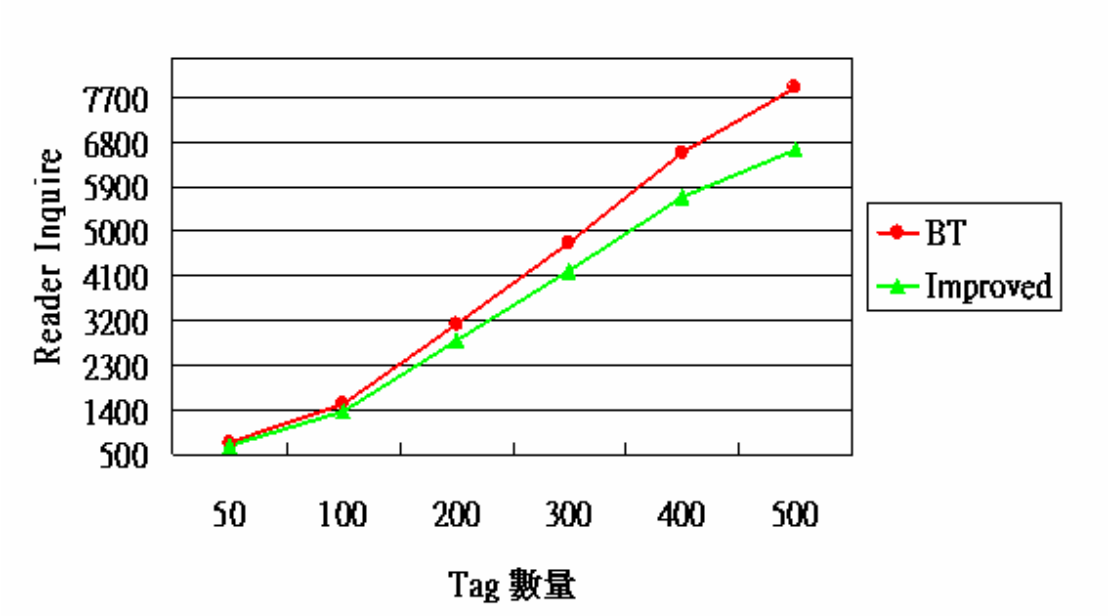

圖 20　針對標籤數量改變的模擬結果(二)

以圖 20 所示的模擬結果而言，在標籤數量較少的時候(200 個標籤以內)，在新式辨識機制下的 Reader 總查詢次數大約和 Binary Tree 辨識架構相同，差異並不大，大約都只有百分之十以內的差距(R2 的值在百分之九十以上)，而當標

籤數量在 200 個以上的時候，新式辨識機制下的 Reader 總查詢次數就開始漸漸地少於 Binary Tree 辨識架構，而當總辨識標籤數量達到 500 的時候，降低 Reader 的總查詢次數大約為十六個百分點(R2 的值約為百分之八十四左右)。

綜合以上的兩個模擬實驗可以觀察到，本論文所提出的新式辨識機制在當標籤數量漸漸增加後，標籤總回應次數的下降幅度愈來愈大，R1 的值從百分之三十二點四慢慢下降到百分之二十六點二；而在 Reader 的總查詢次數上也有著相同的情形，R2 的值從百分之九十點四下降到百分之八十四點一。造成這樣的結果主要是因為當標籤的數量比較多的時候，新式辨識機制所能影響到 Reader 查詢和標籤回應的機率較高，所以才會呈現標籤數量愈多效率愈好的結果。

4.5.3 針對標籤 ID Code 相似度的模擬結果

相似度高的情況

這個部份的模擬實驗主要是要檢驗標籤 ID Code 的相似程度對 Binary Tree 辨識架構以及本論文所提出的新式辨識機制是否有影響，以及影響程度的大小，模擬實驗的過程首先是將標籤的 ID Code 值從 0 開始增加，如果標籤的數量有 50 個，這 50 個標籤的 ID Code 值就是從 0 到 50，標籤的數量如果是 100 個，這 100 個標籤的 ID Code 值就是從 0 到 100，以此類推。這樣的設計可以讓標籤 ID Code 的值相似度達到最高，而下圖 21 即為在各種不同標籤數量下，兩種辨識模式對標籤 ID Code 相似度高的時候所需的標籤總回應數的模擬結果。

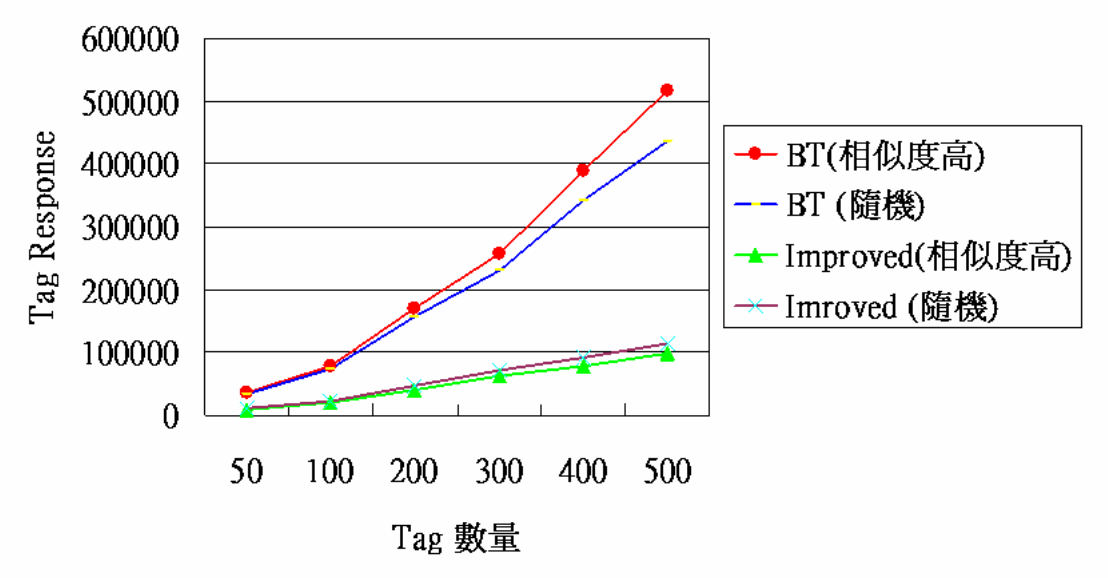

圖 21　針對標籤 ID Code 相似度高的模擬結果(一)

觀察上圖的模擬結果，當標籤 ID Code 值相似度高的時候，Binary Tree 辨識架構所需的標籤總回應次數比原本隨機產生的 ID Code 來得多，增加的幅度大約是六到十五個百分點，隨著標籤數量的增加，回應次數增加的比例也愈高；而在新式辨識機制中，標籤的總回應次數比隨機產生的 ID Code 回應數少，減少的比例大約是在七到十三個百分點之間不等，而標籤的數量愈多，回應次數減少的比例也愈高。

原本的 Binary Tree 辨識模式在標籤 ID Code 相似度高的時候，因為每次符合 Reader 查詢位元的標籤數會變多，所以標籤的回應次數勢必會增加；而新式辨識機制在標籤 ID Code 相似度高的時候，因為符合 Reader 查詢位元的標籤數增加，所以先前儲存的 Reader 查詢資訊就更有效用，標籤的回應次數也就跟著降低了。

相似度低的情況

相對於先前針對標籤 ID Code 相似度高的模擬實驗，本小節則針對標籤 ID Code 相似度低的情況來進行模擬實驗，測試兩種辨識架構在標籤 ID Code 相似度低的情況下所需的總溝通次數是否會受到影響，以及影響的程度大小。

模擬實驗的過程是將標籤的 ID Code 值的分配以最大漢明距離(Hamming

Distance)為原則，每次產生一個新的標籤 ID Code，就從第一個位元開始，和所有已產生出來的 ID Code 做比較，如果現有標籤在該位元 ID Code 的值為 0 的標籤數量較多，那麼新標籤 ID Code 在該位元的值就設為 1，反之，如果現有標籤在該位元 ID Code 的值為 1 的標籤數量較多，那麼新標籤 ID Code 在該位元的值就設為 0，重複相同的步驟直到填滿每一個位元的值為止即完成一個標籤 ID Code 的產生。這樣產生標籤的設計可以讓標籤 ID Code 的值相似度達到最低，而下圖 22 即為在各種不同標籤數量下，兩種辨識模式對標籤 ID Code 相似度低的時候所需的標籤總回應數的模擬結果。

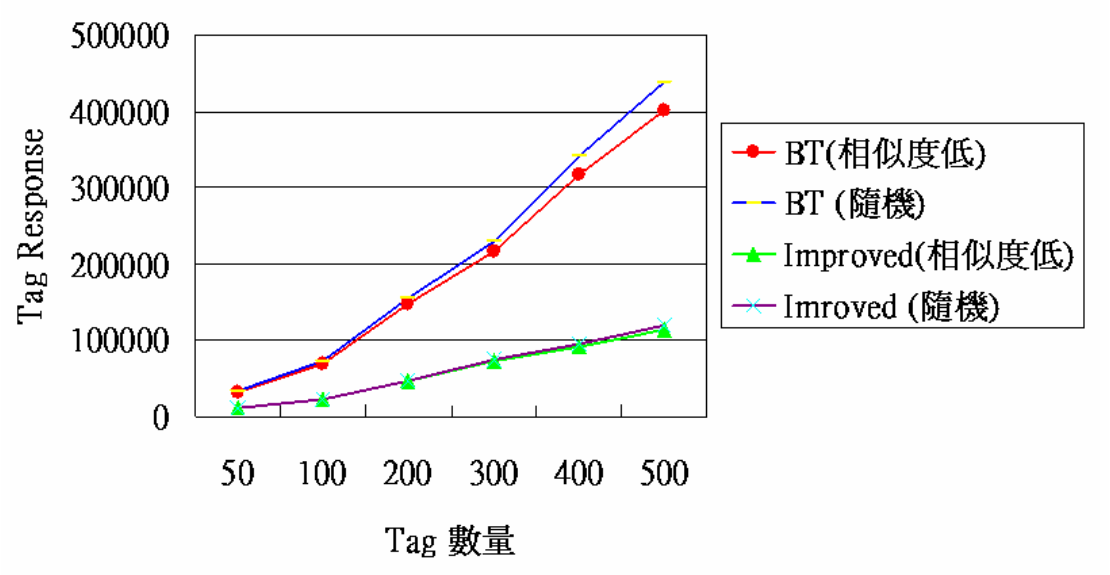

圖 22　針對標籤 ID Code 相似度高的模擬結果(一)

根據上圖的模擬結果顯示，在標籤 ID Code 值相似度低的情況下，Binary Tree 辨識架構所需的標籤總回應次數比隨機產生的 ID Code 少，如果和先前 ID Code 相似度高的情況相比較，兩者的差距就更大了，而回應次數下降的幅度大約是六到十五個百分點；而在新式辨識機制中，標籤的總回應次數有少許的增加，增加的比例大約是在二點五到四點七個百分點之間不等，影響的程度並不大。

原本的 Binary Tree 辨識模式在標籤 ID Code 相似度低的時候，因為每次符合 Reader 查詢位元的標籤數會變少，所造成的標籤回應次數就勢必會減少；而新式辨識機制在標籤 ID Code 相似度低的時候，因為每次符合 Reader 查詢位元的標籤數變少，所以先前儲存 Reader 查詢資訊的效用就變低了，標籤的回應次數也就自然地增加了。

4.5.4 不同記憶體使用量的模擬結果

標籤總回應次數

由於本論文所提出的新式辨識機制需要有額外的記憶體用來儲存先前的查詢資訊，而且當標籤位元長度較長時，不可能將每一次 Reader 的查詢結果都予以儲存，所以使用的記憶體數量必定會有限制，而以下的模擬實驗就是根據新式辨識機制在不同的記憶體使用量下，在不同辨識標籤數量時完成標籤辨識一共需要多少次的標籤回應數，下圖 23 比較了四種不同記憶體使用量的新式辨識機制以及原本的 Binary Tree 辨識架構在各種不同的標籤數量下所需的標籤總回應次數。

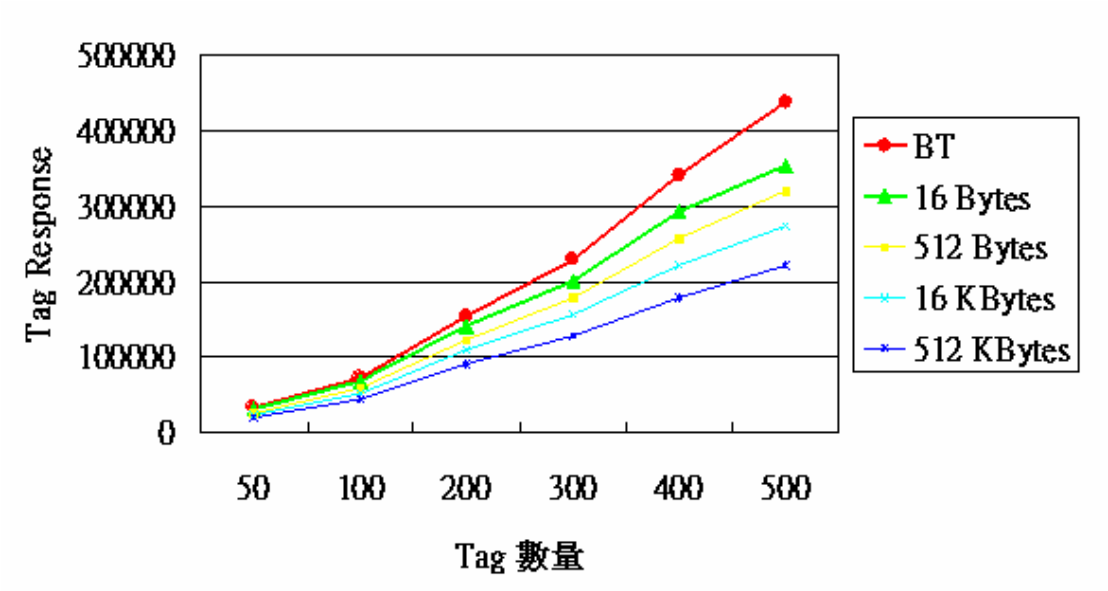

圖 23 針對不同記憶體使用量的模擬結果(一)

根據上圖 23 的模擬實驗結果，可以發現當新式辨識機制所使用在儲存先前查詢結果的記憶體愈多，標籤的總回應次數也會隨之下降，但是下降的幅度卻是呈現遞減的現象，另外，在標籤數量不多的時候，下降的比例也較為有限，而隨著標籤數量的增加，下降的比例也隨之增加。在標籤數量到達 500 個的時候，16 Bytes 的記憶體使用量，使得標籤的總回應次數降為原本 Binary Tree 辨識架構的百分之八十二左右(R1= 0.81)，而當記憶體的使用量到達 512 K 的時候，R1 的值就下降到了百分之五十左右了。

Reader 總查詢次數

不同記憶體使用量的模擬實驗，除了前一小節所討論的標籤總回應次數之外，另外 Reader 的總查詢次數也是一項非常重要的評估指標，在本小節裡，將實驗的縱座標改為 Reader 的總查詢次數，測試 Binary Tree 辨識架構以及新式辨識機制在各種不同的記憶體使用量下所需的 Reader 查詢次數的差異，下圖 24 即為此模擬實驗的數據圖表。

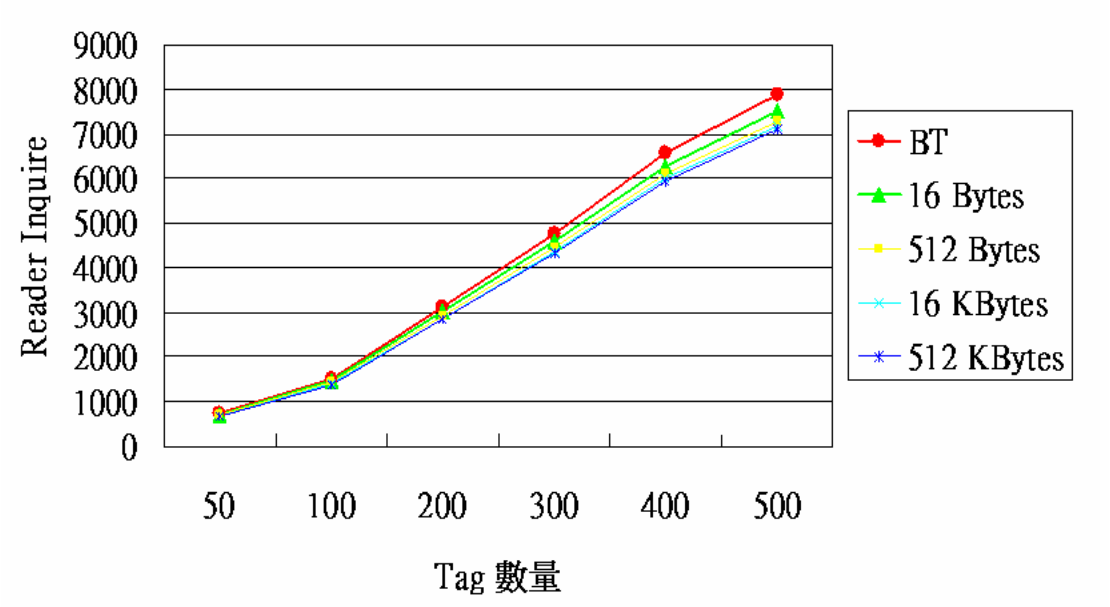

圖 24　針對不同記憶體使用量的模擬結果(二)

根據圖 24 實驗結果，可以發現當新式辨識機制所使用在儲存先前查詢結果的記憶體愈多，Reader 所需的查詢次數也跟著隨之下降，而在標籤數量到達 500 個的時候，16 Bytes 的記憶體使用量，使 Reader 的查詢次數降到原本 Binary Tree 辨識架構的百分之九十五左右(R1= 0.95)，而當記憶體的使用量到達 512 K 的時候，R1 的值就下降到了百分之九十左右。

最後將上述的圖 23 和圖 24 分別轉換成量化的表格數據表 5 和表 6，表格中的百分比數值代表在該標籤數量下，使用某固定大小記憶體的新式辨識機制，在完成辨識後一共需要的溝通次數為原本 Binary Tree 辨識架構的百分之幾，也就是計算不同情況下的 R1 和 R2 的值。

表 5　改善比例表(一)

標籤數量\記體數量	16 Bytes	512 Bytes	16 KBytes	512 KBytes
50	97.1%	81.8%	72.8%	62.2%
100	94.1%	80.9%	71.9%	60.8%
200	90.7%	79.1%	70.5%	58.6%
300	87.8%	77.7%	67.8%	55.9%
400	85.4%	75.5%	65.1%	52.2%
500	82.1%	73.2%	62.5%	50.6%

表 6　改善比例表(二)

標籤數量\記體數量	16 Bytes	512 Bytes	16 KBytes	512 KBytes
50	97.4%	94.5%	92.0%	90.8%
100	97.0%	94.1%	91.8%	90.7%
200	96.5%	93.6%	91.6%	90.6%
300	96.1%	93.2%	91.4%	90.5%
400	95.6%	92.7%	91.2%	90.4%
500	95.1%	92.2%	91.0%	90.3%

4.5.5　綜合分析與結論

綜合歸納以上的各項模擬實驗和數據，可以得到以下的一些結論：

1.　當標籤的位元長度增加時，R1 的值從 10 位元的百分之二十九點五降

到 96 位元的百分之二十七點八，R2 值從 10 位元的百分之九十八點七降到 96 位元的百分之九十六點七，這意味著在 Binary Tree 辨識架構下所需的溝通次數會隨著標籤位元長度而增加，且增幅略大於新式的辨識機制；而在新式辨識機制下所需的溝通次數，增加的幅度就較為平緩。

2. 針對標籤的數量而言，在辨識標籤總數量為 50 到 100 個標籤時，R1 的值大約是在百分之三十到三十二之間，而隨著標籤數量持續增加後，在 Binary Tree 辨識架構下標籤總回應次數的增加幅度要比新式的辨識機制的增加幅度大，到標籤總數量為 500 個的時候，R1 的值就大約降到了百分之二十六左右了；而在 Reader 的總查詢次數上也有著相同的情形，R2 的值從百分之九十點四下降到百分之八十四點一。如果標籤的數量大於 200 個的時候，且記憶體成本在可接受的範圍內，選用本論文所提出的新式辨識機制可能會有較高的辨識效率。

3. 再以標籤的 ID Code 相似度觀之，在標籤的 ID Code 相似度高的情況下，Binary Tree 辨識架構所需的標籤總回應次數比原本隨機產生的 ID Code 來得多，增加的幅度大約是六到十五個百分點，而新式辨識機制在同樣的情況下，總溝通次數反而下降；反之，在標籤的 ID Code 相似度低的情況下，Binary Tree 辨識架構所需要的溝通次數比原本隨機產生的 ID Code 來得少，下降的幅度大約是六到十五個百分點；而在本論文所提出的新式辨識機制中，標籤的總回應次數卻有少許的增加，增加的比例大約是在二點五到四點七個百分點之間不等，這意味著在標籤 ID Code 相似度高的情況下，較適合使用新式的辨識機制；而在標籤 ID Code 相似度低的情況下，新式辨識機制的效用較差，如果有記憶體成本上的考量，則選用 Binary Tree 辨識模式，在效能上也應可接受。

4. 最後，針對新式辨識機制所需的記憶體而言，原則上使用的記憶體數量愈多，所能降低溝通次數的百分比愈高，但是每多增加一單位的記憶體，所能多減少的溝通次數呈現遞減的情形，也就是邊際效益愈來愈低，所以使用的記憶體多寡，必須取決於實際應用上所要求的效率高低，以及對記憶體成本的容許度來決定。

第五章　總結與未來發展

5.1 總結

本論文的主要目的在於提出一個新式辨識機制，使得 Reader 在多 Tag 的辨識環境下能夠有效地降低 Reader 和標籤之間的溝通次數，由於原本 RFID 的基礎辨識架構 Binary Tree Scheme 在標籤數量到達 200 個以上時，Reader 和標籤之間的溝通次數會開始呈現較大幅度的增加，這會造成整個辨識過程的延遲以及整體電力消耗的浪費。

在本論文所提出的新式辨識機制裡，在有效地降低 Reader 和 Tag 之間溝通次數的背後，同時隱含著記憶體成本的付出，但是隨著記憶體使用量的增加記憶體的邊際效益卻呈現遞減的現象，這也意味著過多的記憶體使用，不但無法更進一步有效地改善辨識效率，還會造成記憶體成本的浪費，不過因為記憶體的使用是在 Reader 這一端，所以增加記憶體的使用並不會違反 RFID 便宜以及大量化的原則。

5.2 未來發展

近幾年來快速興起的射頻辨識技術大多都把重點放在硬體設計的研發上，包括小化標籤的大小、控制天線的收訊範圍以及標籤的可重用設計等等，而軟體的部分像是演算邏輯的改進也在最近漸漸受到重視，RFID 的應用相信不論是在硬體或軟體上，都需要有相輔相成的發展才能有更廣泛且全面的未來。

本論文的焦點著重在改善 RFID 單一 Reader 同時辨識多個 Tag 的辨識架構，至於其他的辨識相關領域，像是移動中的標籤辨識、多個 Reader 一起進行辨識工作，或者是混雜多種標籤位元格式的辨識方式，相信都會是未來 RFID 辨識技術發展的重點。

參考文獻

[1] S.Sarma, David L. Brock, Kevin Ashton,“The Networked Physical World”

[2] 鍾惠安,”電子商務導航 - RFID”經濟部資策會

[3] D. Brock “The Electronic Product Code (EPC)”

[4] K. Finkenzeller, “RFID Handbuch”.Carl HANSER Verlag.

[5] A. Sama & M. Balakrishnan, “Speeding up Power Estimation of Embedded Software”.International Symp. on Low Power Design, 2000.

[6] http://www.auto-id.org.Protocol specification for a 900 MHz Class 0 Radio Frequency Identification Tag.

[7] C. Law & K. Lee, “Efficient Memoryless Protocol for Tag Identification”.Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, August 11,2000, Boston, Massachusetts.

[8] F. Zhou, D. Jing , C. Huang & M. Hao, “Optimize the Power Consumption of Passive Electronic Tags for Anti-collision Schemes”. The 5th ASICON, Oct., 2003 Beijing, China.

[9] E. H. L. Aarts, and J. H. M. Korst, Simulated Annealing and Boltzmann Machines. John Wiley & Sons, Chichester, 1989.

[10] A.G. Barto, S. J. Bradtke, and S. P. Singh, Learning to act using real-time dynamic programming. Artificial Intelligence, vol 72, pp. 81-138, 1995.

[11] R. E. Bellman, and S.E. Dreyfus, Applied Dynamic Programming. Rand Corp., 1962.

[12] M. B. Cozzens and F.S. Robert. T-colorings of Graphs and the Channel Assignment Problem. Congressus Numerantium, Vol. 35, pp. 191-208, 1982.

[13] M. Duque-Ant n, D. Kunz, and B. R ber, Channel Assignment for Cellular Radio Using Simulated Annealing. IEEE Transactions on Vehicular Technology, Vol. 42, pp. 14-21, 1993.

[14] D. W. Engels, The Reader Collision Problem. Auto-ID Center, Massachusetts Institute of Technology, 2001.

[15] Back, Thomas “Advanced algorithms and operators ”, 2000