提昇碰撞決議演算法對 802.11 分散協調功能協定的效益

陳俊佑 薛銘鉉 王鵬舟
義守大學
資管所研究生
Chunyuchen1983@gmail.com

危永中
義守大學
資管所助理教授
Wei3504d@isu.edu.tw

摘要

IEEE 802.11 協定是無線區域網路（WLANs）的標準，其在 MAC 層協定所使用的存取方式為分散協調功能（Distributed Coordination Function, DCF）亦就是載波感應多重存取/碰撞避免（CSMA/CA）機制。然而根據此標準的實作，其避免碰撞之二元指數退後演算法（Binary Exponential Backoff Algorithm, BEB）參數是與 WLANs 環境息息相關；尤其是在工作點增加所導致之高負載環境時，由於碰撞率增加或在工作點成功傳輸後之當時媒介狀態的資訊沒有保留，使得競爭視窗（Contention Window，CW）的設定值再次被設定為最小值，致使分散協調功能協定效能逐漸降低。時至今日，已有許多有關運用碰撞決議演算法(collision resolution algorithm)用來增強 DCF 協定效能方面的討論，本研究針對數種碰撞決議演算法，以 C++ 語言分別進行模擬分析比較，就其內容與設計的特點進行深入討論，檢討其未盡完善之處；並提出一個簡單有效的效能提昇方法；其利用於不同競爭階段對競爭視窗範圍進行不重複地切割，進而提高每個競爭階段的最小與最大競爭視窗值，使其兼具考量現有網路負載情形及上一競爭階段歷史記憶特性；根據實際模擬結果顯示，本研究所提出的方法可分別結合運用在數種不同的諸如 BEB，DIDD (Double Increase Double Decrease)，EIED (Exponential Increase Exponential Decrease)等碰撞決議演算法，明顯的提昇與改善分散協調功能協定效能。

關鍵詞： 802.11、流量、延遲、競爭視窗、分散式協調功能

1. 背景簡介

無線區域網路(WLANs)因能提供許多在有線區域網路受到限制的環境下另一種解決方案，而成為今日通訊業新寵。在許多應用科技中我們已可見到其蓬勃發展的身形。未來隨著其頻寬及速率的不斷提升，無線區域網路已成為有線區域網路的一種重要且不可或缺的輔助技術。

無線區域網路(WLANs)所使用的協定為 IEEE802.11 通訊協定，其為 IEEE 委員會所制定專為解決無線區域網路通訊問題的一種規則。其主要內容有下列兩項：

- 設計高效能的無線區域網路之規格
- 與其相關的安全性整合性及服務品質問題

媒體存取控制(MAC)協定為 IEEE802.11 協定中的一個子協定，它主要的工作為負責在實際傳輸媒介上傳遞數據。它是資料鏈結層中負責與實體通信的部分。

IEEE802.11 協定中之媒體存取控制協定主要是設計來使工作點間能分享一個單一媒介。此種協定又稱分散協調功能協定(DCF)。其主要作用為避免在無線區域網路通信的過程中發生因為同時傳輸而造成的通訊障礙。換句話說，即是要執行在碰撞避免偵測(CSMA/CA)功能。此種功能在吾人常用的有線區域網路(此處指乙太網路)中甚為熟悉與常見。然而，當工作點增多時，由於碰撞率增加，分散協調功能協定效能便逐漸降低。如此一來整個無線區域網路的效能將逐漸降低。

針對這個現象，已有許多方法被提出以解決此一無線網路壅塞的問題。這些種方法一般稱為碰撞決議演算法(collision resolution algorithm)。截至目前為止，已有許多種碰撞決議算法被提出，以試圖解決因為當工作點增多致使 DCF 效能逐漸降低的問題。不過，其演算法中皆有未盡完善之處。

本文將藉由實際模擬結果，針對幾種碰撞協議算法作出討論與分析比較，檢討其算演法中之未盡完善之處。同時，我們也將提出我們自己的一個能適用於各種碰撞決議演算法的解決方案，以解決 DCF 協定效能的問題。

本篇論文組織架構簡述如下：第二節將介紹 DCF 協定及 Bianchi 的 DCF 模型。第三節則是針對目前提出的數種碰撞決議演算法作

分析比較，第四節我們將提出我們本身的改善方法，我們稱之為增強分散協調改良(Enhanced Distributed Coordination Improvement, EDCI)方案，第五節將對分佈協調改良方案與其他碰撞決議算法作比較，最後是本篇論文的總結。

2. 分散協調功能(DCF) 與 Bianchi 的 DCF 模型

2.1 分散協調功能(DCF)

在 IEEE 802.11 中主要提供 DCF 與 PCF(Point coordination Function)這兩種協調式功能來進行媒介(channel)的存取。DCF (圖 1) 是其基本的存取方式，主要是以 CSMA/CA 技術進行非同步的傳輸，在 DCF 期間稱之為競爭週期。工作點要在競爭週期取得傳送的機會，必須先去檢查媒介上是否為閒置時間(idle time)，如果媒介上是閒置時間，在等待一段 DIFS(Distributed Interframe Space) 時間後如果媒介上一直處於閒置狀態，則可以立即傳送資料；但假如媒介在等待時間內變為忙碌狀態，工作點必須在下一次閒置時間開始前先等待一段 DIFS 時間，然後進入後退等待時間(backoff time)，其中後退等待時間是由競爭視窗中依據均勻分佈(Uniform Distribution)隨機選取一個介於 CWmin 與 CWmax 之間的競爭視窗值，隨著媒介閒置一個時槽(Slot time)，後退計數器(Backoff time Counter)便會遞減 1 個時槽。在倒數期間若媒介變為忙碌，則後退計數器會暫停倒數，並*凍結*其目前倒數時間，等到下一次後退時間時再繼續倒數，直到後退計數器為 0 時即可取得傳送的機會。如果環境中有另一或多個工作點同時倒數到 0，則會發生碰撞(collision)；在次碰撞發生後，CW 以 BEB 方式增加 CW 長度(Contention Window Size)直到 CWmax，來降低碰撞的機會；而當傳送成功之後，CW 長度將重設回 CWmin。

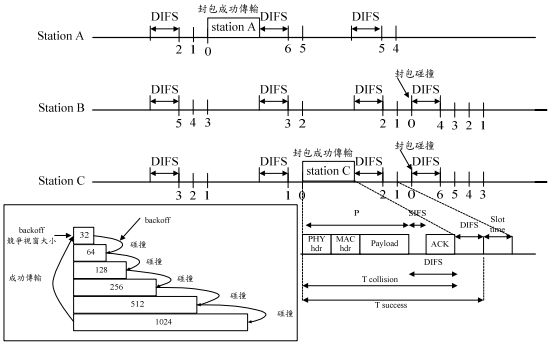


圖 1 BEB 基本存取(basic access) 機制的運作模型

IEEE 802.11 中碰撞避免的後退等待（Backoff）機制是與 WLANs 環境息息相關，尤其是在高負載時發生高碰撞率或在成功的傳輸後，當時媒介狀態的資訊，因沒有保留，而使得 CW 長度再次被設定為初始值而致使媒介媒介利用率下降。此外無線網路中如果有一封包丟失時，送出訊框將會更努力傳送，這時若以延長退後等待的時間來處理，因為會導致頻寬的浪費，將反使得事情更糟糕。為了解決這些現象，已有許多藉由控制競爭視窗增減量，以調適因應 WLANs 環境工作點變化之改善 BEB 的碰撞決議演算法被提出；其中又以 DIDD (Double Increase Double Decrease) (Wen-Kuang Kuo,2003)與 EIED (Exponential Increase Exponential Decrease) (Ivan N. Vukovic,2004)因為僅單純涉及演算法中控制競爭視窗參數(CWmin, CWmax , M)及成功傳送後競爭視窗大小的選擇，而格外受到重視。這些演算法雖可提昇 BEB 碰撞決議演算法的效能，但很可惜的是他們僅考量在高負載環境下，成功傳送後減緩設定競爭視窗變小速度來舒緩高競爭工作點的流量，而忽略在高競爭工作點環境下碰撞後的競爭工作點流量控制。本文以 C++ 模擬針對高負載環境下，增加競爭視窗平均長度的觀念所設計的碰撞決議演算法對 BEB，DIDD，EIED 等三種演算法進行高負載環境下性能評估；根據實際模擬結果顯示，本研究所提出的方法可輕易結合運用在各種不同之演算法上，明顯的提昇與改善分散協調功能協定效能。

2.2 Saturation Throughput & Delay Analysis

本研究採用 Bianchi 所提出的二維馬可夫鏈模型，對不同的碰撞決議演算法，進行模擬分析，其相關參數分析如下:

2.1.1 Saturation Throughput (Bianchi,2000)

假設每一個工作點的碰撞機率 p 是一個常數，且媒介沒有隱藏工作點的問題，在固定的競爭工作點數量 N 之下，每一個工作點在每一次成功的傳送之後，立即會有下一個封包要做傳送。令 $s(t)$ 為一個隨機過程，代表一個工作點在時槽 t 時，它的 backoff 階層(stage)M；令 $b(t)$ 為一個隨機過程，代表一個工作點在時槽 t 時，它的 backoff 競爭視窗大小；定義 $W_i = 2^i W$ ，$i \in (0, M)$ ，W_i 為第 i 階競爭視窗

大小;則二維馬可夫鏈來模擬隨機過程 $\{s(t),b(t)\}$，可以如下圖所示：

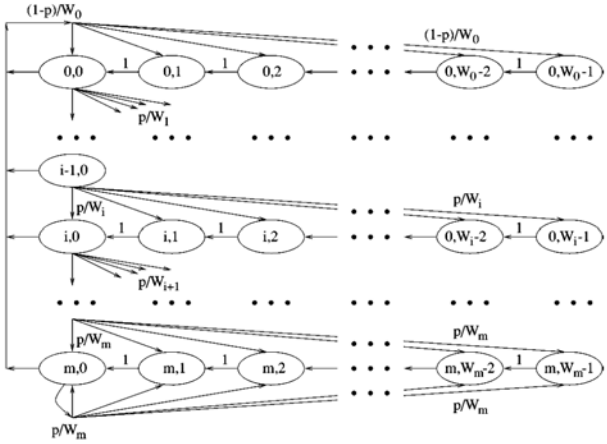


圖 2 馬可夫鏈模型

在這個馬可夫鏈當中，在某種狀態發生的條件之下，下一個狀態會發生的機率如下：

$$\begin{cases} P\{i,k \mid i,k+1\}=1 & k\in(0,W_i-2) \quad i\in(0,M) \\ P\{0,k \mid i,0\}=(1-p)/W_0 & k\in(0,W_0-1) \quad i\in(0,M) \\ P\{i,k \mid i-1,0\}=p/W_i & k\in(0,W_i-1) \quad i\in(1,M) \\ P\{M,k \mid M,0\}=p/W_M & k\in(0,W_M-1) \end{cases} \tag{1}$$

在(1)中，第一個方程式說明了，在每一個時槽起始的時候，它的後退等待時間就已經被決定了。第二個方程式說明了，在一個成功的封包傳送之後，下一個新的封包它會從第 0 階開始，因此它的後退等待時間是均勻地從 $(0,W_0-1)$ 這個範圍被選出來。其他的方程式則表示這個系統發生碰撞的情形。第三個方程式說明了，當碰撞發生在第 $i-1$ 階時，則階層數 $s(t)$ 增加，且新的初始後退等待時間是均勻地從 $(0,W_i\text{-}1)$ 這個範圍被選出來。第四個方程式說明了，因為階層數 M 到達最後一階，所以會一直停留在最後一階不斷地選擇後退等待時間。

設 $b_{i,k}=\lim_{t\to\infty}P\{s(t)=i,b(t)=k\}, \quad i\in(0,M),$

$k\in(0,W_i-1)$ 為此馬可夫鏈的穩定過程。我們可以對這個馬可夫鏈得出一個封閉形式的式子如下：

$$\begin{aligned} b_{i-1,0}\cdot p=b_{i,0} &\longrightarrow b_{i,0}=p^i b_{0,0} \quad 0<i<M \\ b_{M-1,0}\cdot p=(1-p)b_{M,0} &\longrightarrow b_{M,0}=\frac{p^M}{1-p}b_{0,0} \end{aligned} \tag{2}$$

由於馬可夫鏈的規律性，對於每一個 $k\in(1,W_i-1)$ 來說，可以表示成：

$$b_{i,k}=\frac{W_i-k}{W_i}\cdot\begin{cases}(1-p)\sum_{j=0}^{M}b_{j,0} & i=0\\ p\cdot b_{i-1,0} & 0<i<M\\ p\cdot(b_{M-1,0}+b_{M,0}) & i=M\end{cases} \tag{3}$$

根據(2)的關係，且利用 $\sum_{i=0}^{M}b_{i,0}=b_{0,0}/(1-p)$，可以將(3)改寫成：

$$b_{i,k}=\frac{W_i-k}{W_i}b_{i,0} \quad i\in(0,M), \quad k\in(0,W_i-1) \tag{4}$$

因此，透過(2)跟(4)的關係，$b_{i,k}$ 的全部數值被表示成數值 $b_{0,0}$ 的函式與條件碰撞機率 p 的函式。$b_{0,0}$ 最後會被簡化成(5)，表示如下：

$$\begin{aligned} 1&=\sum_{i=0}^{M}\sum_{k=0}^{W_i-1}b_{i,k}=\sum_{i=0}^{M}b_{i,0}\sum_{k=0}^{W_i-1}\frac{W_i-k}{W_i}=\sum_{i=0}^{M}b_{i,0}\frac{W_i+1}{2} \\ &=\frac{b_{0,0}}{2}\left[W\left(\sum_{i=0}^{M-1}(2p)^i+\frac{(2p)^M}{1-p}\right)+\frac{1}{1-p}\right] \end{aligned} \tag{5}$$

其中 $b_{0,0}$ 為一個穩定的機率分佈，代表經過一段很長的時間之後，$\{s(t),b(t)\}$ 在 $s(t)=0$、$b(t)=0$ 這個狀態下的機率，如下(6)所示：

$$b_{0,0}=\frac{2(1-2p)(1-p)}{(1-2p)(W+1)+pW\left(1-(2p)^M\right)} \tag{6}$$

之後我們便可以將在某一個隨機選取時槽發射封包的機率 τ 表示成：

$$\tau=\sum_{i=0}^{M}b_{i,0}=\frac{b_{0,0}}{1-p}=\frac{2(1-2p)}{(1-2p)(W+1)+pW\left(1-(2p)^M\right)} \tag{7}$$

由於 τ 值是根據條件碰撞機率 p 來決定的，為了找出 τ 值，我們必須先求出碰撞機率 p；碰撞機率 p 代表在 $(n-1)$ 個工作點中，至少會有一個工作點傳送封包的機率，即某一工作點傳送封包，結果發生碰撞的機率：

$$p=1-(1-\tau)^{N-1} \tag{8}$$

設 P_{tr} 為在考慮的時槽中，至少有一個傳送的機率，如下(9)所示：

$$P_{tr}=1-(1-\tau)^{N} \tag{9}$$

而 P_s 為至少有一工作點可以傳送的條件之下，僅只有一工作點傳送的機率，則：

$$P_s=\frac{N\tau(1-\tau)^{N-1}}{P_{tr}}=\frac{N\tau(1-\tau)^{N-1}}{1-(1-\tau)^{N}} \tag{10}$$

根據 Throughput 的定義：

$$S=\frac{E[\text{payload information transmitted in a slot time}]}{E[\text{length of a slot time}]} \quad (11)$$

其中，$E[\text{payload information transmitted in a slot time}]$ 為在單位時間長度裡面，平均傳送出去的資料封包量，$E[\text{length of a slot time}]$ 為平均時槽的長度。$E[P]$ 為平均資料封包大小；封包資訊在某一個時槽成功被傳送的平均值為 $P_{tr}P_sE[P]$，其中，$P_{tr}P_s$ 為封包在某一個時槽的成功傳送機率；$(1-P_{tr})$ 表示沒有工作點在傳送封包的機率；P_{tr} 表示至少有一個工作點傳送封包的機率；$(1-P_s)$ 表示至少有一個工作點可以傳送封包的條件之下，包含兩個以上的工作點傳送封包的機率。因此，(11)可以寫成如(12)所示：

$$S=\frac{P_sP_{tr}E[P]}{(1-P_{tr})\sigma+P_{tr}P_sT_s+P_{tr}(1-P_s)T_c} \quad (12)$$

其中，T_s 為因為封包傳送成功，媒介被偵測出為忙碌的平均時間；T_c 為因為封包傳送失敗，媒介被偵測出為忙碌的平均時間，σ 為一個閒置時槽的時間。

設 $H=\text{PHY}_{\text{hdr}}+\text{MAC}_{\text{hdr}}$ 為封包的標頭，δ 為傳輸延遲，則考量基本存取模式(13)如下所示：

$$\begin{aligned}T_s^{bas}&=H+E[P]+\text{SIFS}+\delta+\text{ACK}+\text{DIFS}+\delta\\ T_c^{bas}&=H+E[P^*]+\text{DIFS}+\delta\end{aligned} \quad (13)$$

其中，$E(P^*)$ 為在碰撞的情形下，最長封包的平均長度。

2.1.2 Delay Analysis (P. Raptis,2005)

在延遲問題方面，一個計算平均封包延遲的模型(Chatzimisios)可表示如下：

$$E_C[D]=\sum_{i=0}^{M}(E[X_i]\cdot k_i) \quad (14)$$

其中，X_i 為在第 i 階時，一個成功傳輸封包延遲的時間；k_i 為一個成功傳輸封包抵達 i 階的機率。

$E[X_i]$ 如下(15)所示：

$$E[X_i]=d_i\cdot E[\text{length of a slot ime}] \quad i\in[0,M] \quad (15)$$

其中，d_i 為封包在第 i 階時的平均延遲時槽數；$E[\text{length of a slot ime}]$ 為平均時槽數；d_i 與 k_i 如下(16)、(17)所示：

$$d_i=(W_i+1)/2 \quad i\in[0,M] \quad (16)$$

$$k_i=\frac{p^i-p^{M+1}}{1-p^{M+1}} \quad i\in[0,M] \quad (17)$$

由(14)、(15)、(16)、(17)可得到平均封包延遲 $E_C[D]$，如(18)所示：

$$E_C[D]=E[\text{length of a slot ime}]\cdot\sum_{i=0}^{M}\left(\frac{W_i+1}{2}\cdot\frac{p^i-p^{M+1}}{1-p^{M+1}}\right) \quad (18)$$

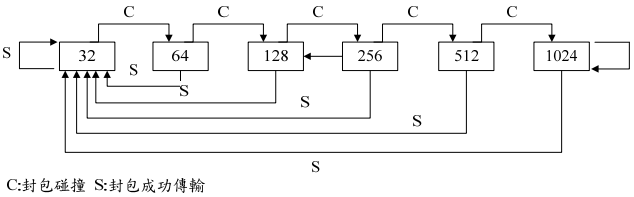
3. 碰撞決議演算法分析比較

3.1 BEB

根據 IEEE 802.11 basic access 模式，如果一個封包可以傳輸，它必須要先去偵測媒介是否為閒置的，如果媒介是忙碌的，這個工作點將延遲發射直到下一個 DIFS 時間被偵測到，然後在傳輸前先等待一個隨機後退時間，來降機碰撞機會。

這個 backoff 時間的計時器只要偵測到媒介是在閒置的狀態會以時槽時間為單位而遞減。這個計時器當媒介是忙碌的時候會停止，而當它會偵測到媒介再次的閒置且大於一個 DIFS 時間後會恢復倒數。

一個工作點的 backoff 計時器到達零的時候將傳送封包。如果目的端成功的接收到封包，它會等待一個短的訊框間區間（SIFS）時間然後回傳一個確認訊號（ACK）。如果傳送端在一個特定的確認訊號截止區間時間內沒有接收到確認訊號，這個封包會被假設為遺失且工作點將會排程而再傳送一次直到成功傳輸為止。



C:封包碰撞 S:封包成功傳輸

圖 3 BEB：CWmin=32 , CWmax=1024

3.2 DIDD

BEE 在一次成功的封包傳輸之後會將競爭視窗大小回復初始狀態。這樣的動作會使得碰撞的機率增加，相對的效能則降低不少。然而此現象在工作點數量很少的時候，是感覺不出其差別的。不過當競爭的工作點數變多的時候，由成功傳輸而帶來的競爭視窗急速縮小導致的碰撞機率急速增加則會大幅的降低效能。

既然 BEB 有上述的情形發生，文獻裡提出一個使競爭視窗大小減少速度較慢的演算法，叫做 DIDD(Double Increase Double decrease)演算法。

DIDD 的主要原理是競爭視窗會在一次成功的封包傳輸後和緩的降低。另一方面與 BEB 相同的是，DIDD 也會在碰撞後將競爭視窗加倍以降低封包的碰撞機率。

DIDD 在封包成功傳輸時，會將競爭視窗減少為成功傳輸前的一半，並不會直接變成為最小競爭視窗，藉此來避免未來可能的封包碰撞。

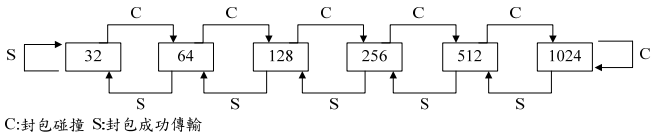


圖 4 DIDD：CWmin=32 , CWmax=1024

3.3 EIED

EIED 是一種相當有彈性的碰撞決議演算法，且具有一定數量的可調整參數。最理想的參數值由網路中運作的條件而決定，比如說網路的負載程度、封包長度等等。在 EIED 中，當一個封包的傳輸因為碰撞而被拒絕時，其競爭視窗將隨著延遲變數 r_I 而增加；而如果此封包成功傳輸，其競爭視窗將隨著延遲變數 r_D 而減少。其運作步驟為:

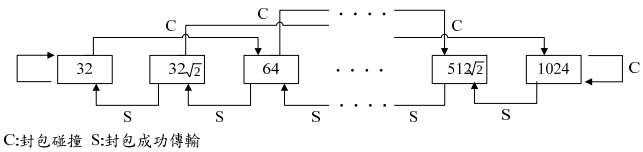


圖 5 EIED：r_I =2, r_D =$\sqrt{2}$ CWmin=32 , CWmax=1024

輸入參數：$\langle M, \mathrm{N}, CW_{\max}, CW_{\min} \rangle$，其中 M 為階層數，N 為工作點數，$CW_{\min}$ 為最小競爭視窗，$CW_{\max}$ 為最大競爭視窗。

Step1.
求出 r_I 值：

$$r_I = \left(\frac{CW_{\min}}{CW_{\max}} \right)^{\frac{1}{M}} \text{，} r_I > 1 \text{，} M \geq 1 \tag{18}$$

Step2.
求出 r_D 值：

$$r_D = (r_I)^{\frac{1}{\mathrm{N}}} \text{，} r_D > r_I > 1 \text{，} \mathrm{N} \geq 1 \tag{19}$$

Step3.
決定競爭視窗 CW

$CW = \min[r_I.CW, CW_{\max}]$，在碰撞的情況下。

$CW = \max[CW / r_D, CW_{\min}]$，在成功傳送的情況下。 (20)

例如：

如果 $r_I \neq r_D$ 且

$M = 12$，$CW_{\max} = 1024$，$CW_{\min} = 16$

則可得 $r_I = 2$，因此 $r_D = \sqrt{2}$

如果 $r_I = r_D$ 且 $M = 6$，$CW_{\max} = 1024$，$CW_{\min} = 16$

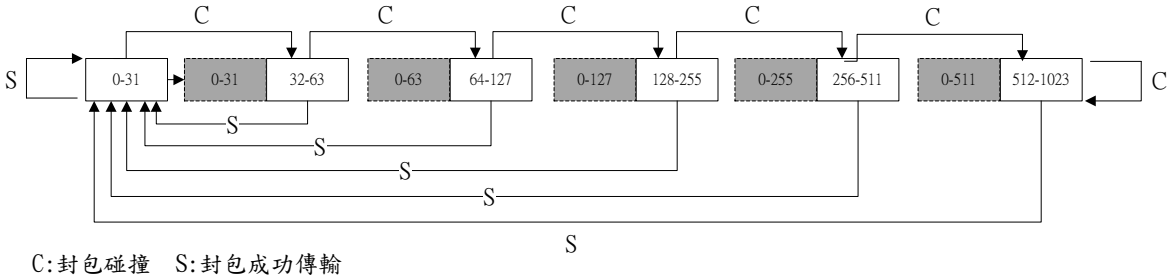
則可得 $r_I = 2$，因此 $r_D = 2$

所以如果在 $r_I = r_D$ 的情況下，EIED 演算法就等於 DIDD 演算法

也就是說 DIDD 演算法算是 EIED 演算法中的一個特例

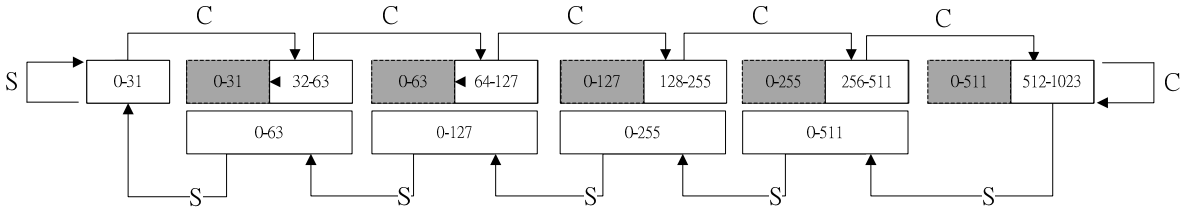
3.4 EDCI (Enhanced Distributed Coordination Improvement)

EDCI 利用於不同競爭階段對競爭視窗範圍進行不重複地切割，進而提高每個競爭階段的最小與最大競爭視窗值，使其兼具考量現有網路負載情形及上一競爭階段歷史記憶特性(如下圖所示)；



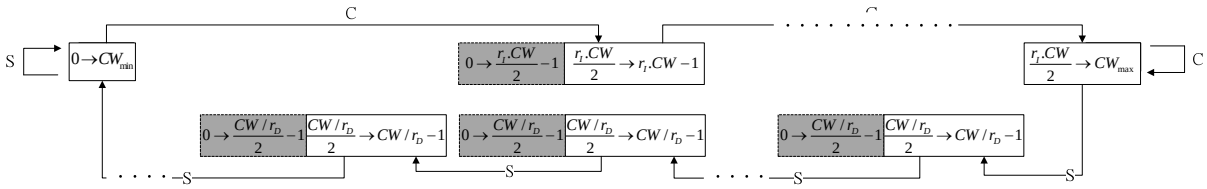
C:封包碰撞　S:封包成功傳輸

圖 6　BEB 使用單向 EDCI



C:封包碰撞　S:封包成功傳輸

圖 7　DIDD 使用雙向 EDCI



C:封包碰撞　S:封包成功傳輸

圖 8　EIEE 使用雙向 EDCI

競爭視窗 CW

$$CW = \min\left[\frac{r_I.CW}{2} \to r_I.CW - 1, CW_{\max}\right],$$

在碰撞的情況下。

$$CW = \max\left[\frac{CW/r_D}{2} \to CW/r_D - 1, CW_{\min}\right],$$

在成功傳送的情況下。

4. 模擬結果分析

本研究以 C++對四種不同演算法進行性能評估與分析，其相關系統模擬參數值設定如表 2。

表 1　系統模擬參數值

DSSS	
Channel bit rate, C	1 Mbit/s
Packet Payload	8184 bits
MAC header	224 bits
PHY header	192 bits
ACK	112 bits + PHY header
Propagation delay, δ	1 us
Slot time, δ	20 us
SIFS	10 us
DIFS	50 us
Minimum CW, W_0	32
Number of CW sizes, M	5

本研究針對三種演算法(BEB、DIDD 和 EIED)提出兩種改良方式，藉由模擬流量(Throughput)、延遲時間(Delay time)、碰撞機率(Collision probability)和階層使用率等構面來探討新型的演算法是否能改善原有演算法的效能。

4.1 流量分析 Throughput Analysis

圖 9 表示 BEB 和 EDCI 的流量分析，EDCI 相對於 BEB 能有效提升流量，且隨著工作點數的增加趨勢更加明顯，但在大量的工作點數量中 EDCI 提升的比率趨於穩定。圖 10 表示在 DIDD、EDCI 單向與 EDCI 雙向三者流量差異，EDCI 單向與 EDCI 雙向都能提昇原有演算法的流量，且隨著工作點數的增加趨勢更加明顯，其提升的比率也趨於穩定，另外 EDCI 雙

向比 EDCI 單向流量提升的幅度不大，此亦表示採用單向 EDCI 與雙向 EDCI 對流量的提昇效果並不顯著。圖 11 表示在 EIED、EDCI 單向與 EDCI 雙向三者流量差異，在少量的節點數中 EDCI 單向和 EDCI 雙向並不能相對有效提升流量，存在大量節點時 EDCI 單向和 EDCI 雙向的流量效益才逐漸彰顯，EDCI 雙向更是明顯。綜合以上三張圖我們可發現這三種演算法的兩種改良方式都可以改善其流量，唯有 EIED 的改良方法必須在大量的工作點數流量才能優於 EIED；圖 12 我們比較 BEB 與結合 EDCI 的 BEB、DIDD、與 EIED 之流量，使用新的改善方法都能比 BEB 相對有效提升流量，其中 EEIED 的流量更是明顯高於圖中各種演算法，這跟原型的 EIED 在封包碰撞或成功降緩改變競爭視窗的速度有密切關聯。

4.2 延遲時間分析 Delay Time

圖 13 表示 BEB 與 EDCI 單向的封包延遲時間，EDCI 單向能相對有效降低封包的延遲時間，隨著工作點數的增加趨勢更加明顯;圖 14 表示 DIDD、EDCI 單向與 EDCI 雙向的封包延遲時間，EDCI 單向比 DIDD 相對有效降低封包延遲時間，而 EDCI 雙向雖能比 DIDD 相對有效降低封包延遲時間，但 EDCI 雙向比 EDCI 單向所能降低的封包延遲時間有限。圖 15 表示 EIED、EDCI 單向與 EDCI 雙向的封包延遲時間，雖然兩種改良的演算法在大工作點數能有相對降低封包的延遲時間，但降低的幅度不大。綜合以上三個圖，我們可看出隨著工作點數增加其封包延遲時間隨之上升，三種演算法的兩種改良方式雖都能相對有效降低其封包的延遲時間，但僅在多工作點數才有差異且降低的幅度都不甚明顯；圖 16 比較 BEB 與結合 EDCI 的 BEB、DIDD、與 EIED 之封包傳遞延遲時間，使用我們 EDCI 方法，在多工作點數環境下，都能進一步改善原有之 BEB、DIDD、EIED 演算法的延遲時間效能。

4.3 碰撞機率(Collision probability)

圖 17 比較 BEB 與結合 EDCI 的 BEB、DIDD、與 EIED 的碰撞機率，每種改良後的演算法都能相對有效降低碰撞機率，而 EDCI 結合 EIED 可有效改善降低碰撞機率隨工作點數量增加所導致之碰撞機率提昇影響。

4.4 階層使用率(Stage Usage rate)

針對階層使用率，我們選取四種工作點數（10、20、40 和 60）來記錄 BEB 與結合 EDCI 的 BEB、DIDD、與 EIED 階層使用情形，從圖 18、19、20 和 21 中可觀察出不管是哪種演算法，在少工作點數中階層數越小使用率越高，BEB、EBEB 和 EIEED 僅在第 0 階就達到將近 70% 的使用率，第 3 階後階層使用率就非常低；這說明在少量的工作點中碰撞機率不高，因此使用的階層數範圍並不大，圖 19、20 和 21 所表示工作點數 20、40 和 60 階層使用率與圖 18 比較可發現隨著節點數增加，第 0 階與第 1 階的使用比例逐漸降低，而後面的階數隨著工作點數增加而使用率也逐漸提高，這代表節點數的增加導致碰撞情況加劇，因此碰撞後移向高階層的機率也逐漸提高。

綜合以上的模擬結果，本研究針對三種演算法所提出的 EDCI 改良方式在流量方面可以相對提升原有演算法的流量，從圖 12 和圖 16 中 EIED 應用 EDCI 流量仍明顯高於 BEB 和 DIDD 應用 EDCI，封包的延遲時間 EIED 應用 EDCI 流量仍明顯高於 BEB 和 DIDD 應用 EDCI，這說明新的演算法僅能提升原演算法的效能，真正能大量提升流量的關鍵因素仍是緩和封包成功傳輸或發生碰撞時競爭視窗大小改變的速度；另外模擬結果亦顯示各演算法的流量分析圖與碰撞機率圖呈現的走勢有高度的相關性，因此封包的碰撞機率與流量具有密切的關係，當碰撞機率低有較高流量現象，反之亦然；所以 EDCI 方法所帶來的成功降低碰撞機率效應，顯然提升了流量且封包的延遲時間不至於高於原形的演算法，確實能改善相對有效提升 DCF 協定效能。

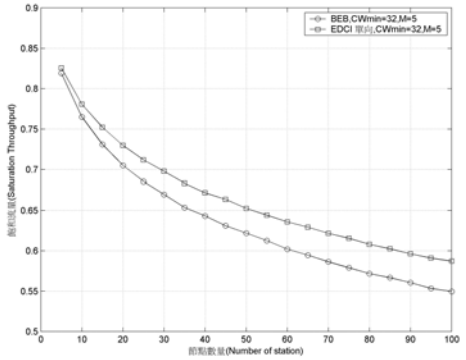


圖 9 Throughput analysis of EDCI and BEB

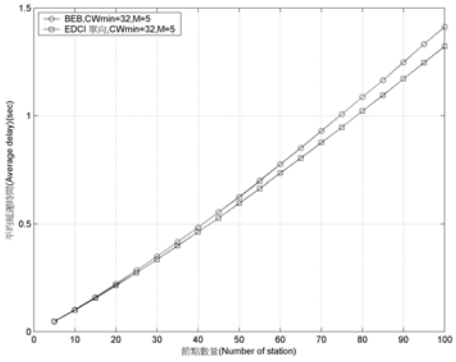


圖 13 Delay analysis of EDCI and BEB

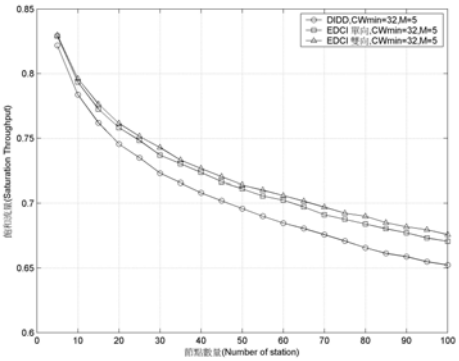


圖 10 Throughput analysis of EDCI and DIDD

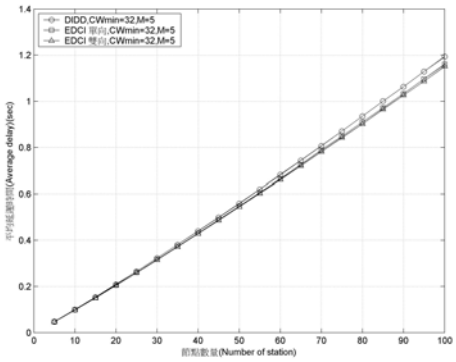


圖 14 Delay analysis of EDCI and DIDD

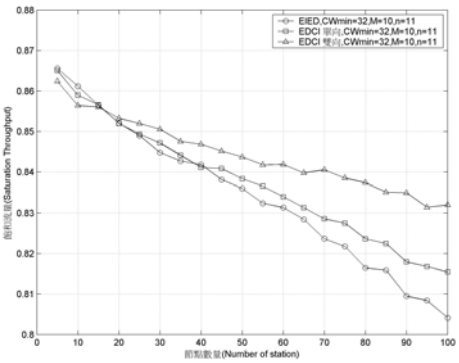


圖 11 Throughput analysis of EDCI and EIED

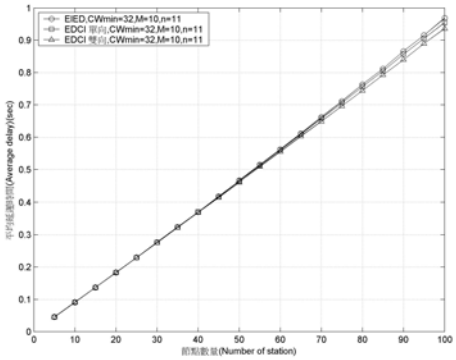


圖 15 Delay analysis of EDCI and DIDD

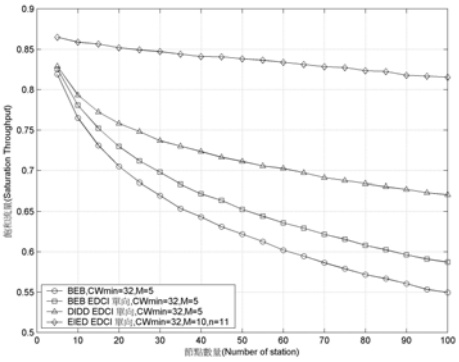


圖 12 Throughput analysis of EDCI 、BEB 、DIDD and EIED

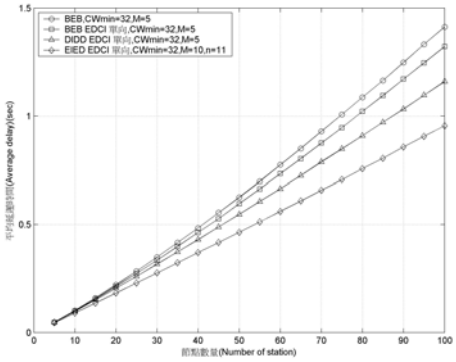


圖 16 Delay analysis of EDCI and DIDD

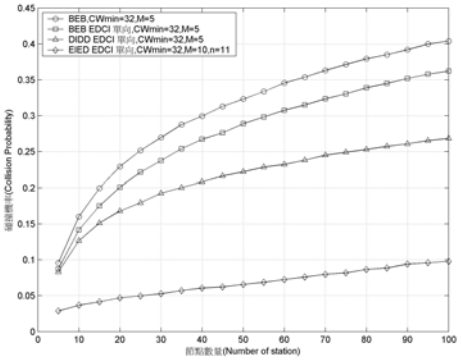


圖 17　碰撞機率比較

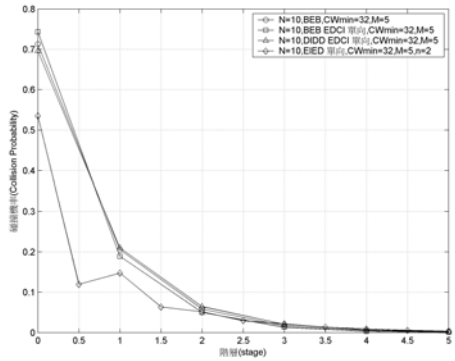


圖 18　階層使用率(N=10)

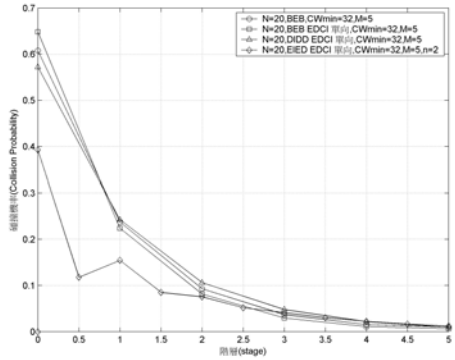


圖 19　階層使用率(N=20)

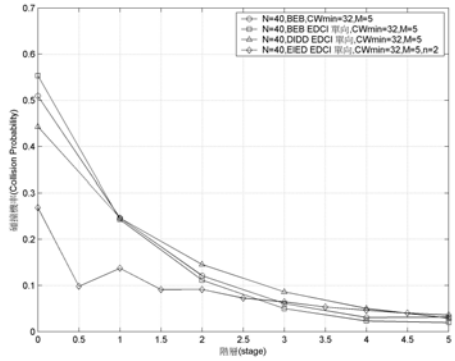


圖 20　階層使用率(N=40)

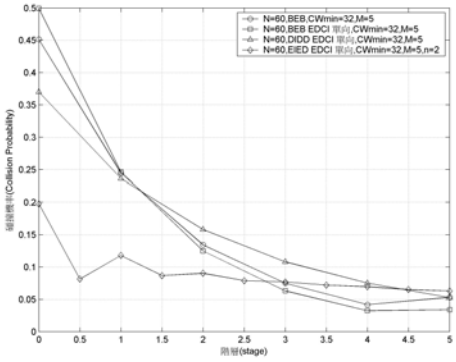


圖 21　階層使用率(N=60)

5. 結論

由模擬結果顯示，EDCI 在流量和延遲時間上都明顯優於 BEB、DIDD 與 EIED，尤其在工作點數量很大的情況下。因此我們可以得知就碰撞決議演算法而言，我們可以透過 $CW_{\max}$、$CW_{\min}$、M 等參數之 EDCI 方法來調整 DCF 協定的效能。

就模擬結果顯示，如果在相同的無線網路環境下，EIED 碰撞決議演算法的運作過程中，如果適當的減少碰撞後競爭視窗的大小，同時在成功傳輸後減緩競爭視窗減少的速度，對於流量與延遲時間的改善會非常明顯；而如果成功傳輸之後的競爭視窗大小減少的越慢，效能同樣會越好。因此如果以後要繼續朝此方向研究的話，可以考慮將此次的研究心得加入 EDCI 的架構中，相信應該會有所幫助。

在這篇文章中，我們針對幾種碰撞決議算法作討論，並作一番分析比較，檢討其算法中之未盡完善之處。同時，我們也提出我們自己的解決方案，試圖解決分佈協調作用協定效能的問題。我們認為我們所提出的分佈協調改良方案，在某種程度上可解決，至少減輕，DCF 協定效能的問題。

參考文獻

[1] G. Bianchi(1998),IEEE 802.11—Saturation Throughput Analysis,IEEE COMMUNCATIONS LETTLERS, VOL. 2, NO.12, DECEMBER.

[2] P. Raptis(2005),Packet Delay Modeling of IEEE 802.11 Wireless LANs.

[3] G. Bianchi(2000), Performance Analysis of the IEEE 802.11 Distributed Coordination Function”IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIOINS, VOL. 18,NO.3,MARCH 2000.

[4] P. Chatqimisios(2003), IEEE 802.11 Packet Delay-A Finite Retry Limit Analysis.

[5] Ivan N. Vukovic(2004), Delay Analysis of Different Backoff Algorithms in IEEE 802.11.

[6] Nah-Oak Song(2003), Enhancement of IEEE 802.11 Distributed Coordination Function with Exponential Increase Exponential Decrease Backoff Algorithm.

[7] Nah-Oak Song(2005), Analysis of EIED Backoff Algorithm for the IEEE 802.11 DCF.

[8] Wen-Kuang Kuo(2003), Enhance Backoff Scheme in CSMA/CA for IEEE 802.11.

[9] P. Chatzimisios(2005), A Simple and effective backoff scheme for the IEEE 802.11 MAC protocol.

[10] Zhifei Li(2005), Performance Analysis of IEEE 802.11 DCF : Throughput, Delay, and Fairness.

[11] Ivan N. Vukov c(2004), Delay Analysis of Different Backoff Algorithms in IEEE 802.11.