Wiki 環境中編輯衝突之研究
A Study of Edit Conflict in Wiki Environment

陳謙民　林敏勝　王邱凱
國立台北科技大學電機工程學系
t5319011@ntut.edu.tw　mslin@ee.ntut.edu.tw　s4318046@ntut.edu.tw

摘要

Wiki(維基)這個詞彙在夏威夷語裡是「快」的意思，被用來當作 Wiki 網站的精神。在 Wiki 的世界裡，每個人都可以快速地修改任何頁面，即使是編輯或是刪除別人寫的文字。由於 Wiki 是一共同寫作的環境，所以在使用者編輯上會有產生編輯衝突的可能性。在此篇論文中，我們以 Diff3 演算法為基礎，加上使用 Ajax 的技術，讓使用者在編輯 Wiki 時，能夠即時的知道是否有編輯衝突的發生，以便提早處裡編輯衝突的問題。
關鍵詞：維基，編輯衝突

Abstract

"Wiki" this word in Hawaii-language means "quick", and the spirit of Wiki website also is "quick". A wiki is a web application designed to allow multiple authors to edit or remove contents straightforwardly. Wiki is a kind of collaborative authoring environment, so edit conflicts would occur for authors. In this paper, we will use Diff3 algorithm and Ajax tool to make the author aware of edit conflicts as soon as possible.
Keywords: Wiki, Edit Conflict

1. 前言

Wiki 是一種應用在網際網路上，開放讓許多人共同創作的一個超文本系統，是由沃德・坎寧安(Ward Cunningham)於 1995 年所建立[4]。基本上 Wiki 網站是開放給任何造訪者，並提供造訪者可以任意的對網站內容作新增、移除的動作，甚至在某些系統中，連登入的動作都不用，因此在這種環境中，使用者編輯的空間是很自由的[1- 3, 5, 6, 8, 11]。

當使用者在 Wiki 系統環境做編輯時，可能會發生數種情況：第一種情況(圖 1)，也就是使用者在做編輯的動作時，並沒有其他使用者針對相同的版本內容做編輯，直到此使用者編輯完畢之後，才有下一個使用者要做編輯，此時未發生編輯衝突的問題；第二種情況(圖 2)，假設使用者 1 在編輯期間（原始版本為 V），有其他使用者 2 對同一個條目做編輯（原始版本也是 V），但是當使用者 1 儲存時（產生新的版本 V1），使用者 2 仍在編輯的狀態，則對使用者 1 就沒有發生編輯衝突的問題，但對使用者 2 而言，因為使用者 1 作儲存的動作，所以造成目前伺服器端所儲存的最新版本已經不再是 V，而是版本 V1，所以對使用者 2 發生了原始版本不同的編輯衝突。此時，引發了以下問題，例如在圖 2 中，假設使用者 1 已經對版本 V 作了需多有價值的編輯貢獻，存檔成版本 V1，但是對於使用者 2 而言，卻無法獲知這些有用的資訊；此外，當使用者 2 編輯完成，存檔成版本 V2 時，如果沒有檢查的機制，此時便會將使用者 1 所完成的版本 V1 直接覆蓋成 V2 版本，相當於忽略了使用者 1 的編輯貢獻。

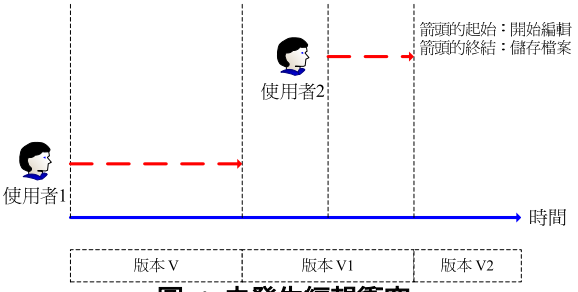

圖 1 未發生編輯衝突

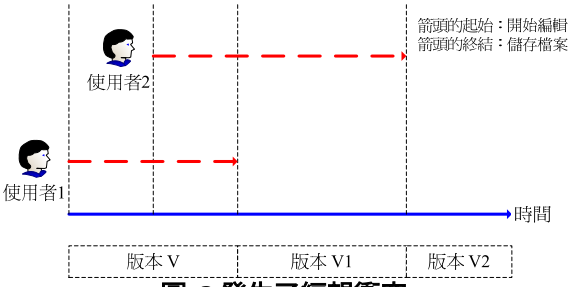

圖 2 發生了編輯衝突

目前解決 Wiki 編輯衝突[7]的方法有：(1)鎖定頁面(Lock the original)[9]：使用鎖住部分資料的方式，同一個時間只允許一個使用者編輯同一區段；(2) 不允許直接覆蓋新的版本(Don't overwrite new revisions)[10]：使用者存檔時提醒該使用者是否有編輯衝突的發生，若有則交給該使用者來手動決定做最後的合併；(3)贏家能得到全部(Winner takes it all)：最後編輯存檔的人當作最新版本，直接覆蓋之前的版本，不管是否有編輯衝突發生與否；(4)使用自動合併的方法(Use an Automatic Merge to reconcile revisions)[12]：若使用者存檔發現有編輯衝突的問題，交由伺服器端自動做合併的動作。

由上述提出的幾個方法可以清楚的知道，對於 Wiki 這一個開放式的自由創作空間，要如何在一個

優質的網路創作空間，提供所有使用者一個良好的編輯環境，已經提出了許多方法來應用，但上述的方法仍有部分的問題，例如：有些方法會採用「鎖定」以及「時間控管」的機制，但是時間的長短則是須仔細考量，畢竟在某些部分的編輯會需要較長的時間，甚至有些使用者可能在編輯的途中，離開了座位，而這段時間內，若有其他使用者欲加入編輯的行列，就會產生問題。因此要如何提供較適合的方法，即是本篇論文所思考的。

Diff3 演算法[14-16]是 Unix 系統中常用的檔案內容比較工具之一，除了比較檔案內容差異之外，Diff3 也可以將所比較的檔案進行合併的動作，當要做合併的時候，我們必須有三個檔案 O、A、B，而 A、B 都是根據同一個原始版本 O 編輯而來的。由於本論文中所應用的環境是在 Wiki 這樣一個開放式的編輯環境，所以可能會有許多人同時在線上編輯的情況，為了減少伺服端的負擔，所以此篇論文 Diff3 的演算法是以 JavaScript[17]編寫在各用戶端上(瀏覽器所瀏覽的網頁中)，當用戶端收到伺服器端所傳送過來的不同版本的內容時，則由用戶端執行 Diff3 演算法，做版本內容差異比較與合併的動作，有此構想主要是為了避免在許多使用者同時編寫時，造成伺服器端的負擔過重。

Ajax[13]是 Asynchronous JavaScript And XML 的縮寫，這是一種使用 JavaScript 進行非同步通訊，交換 XML 資料的技術。在傳統的網頁介面中，經常要讓使用者等待，也就是當用戶端向伺服器端提出請求之後，使用者所能做的就是等待伺服器端的回應；就是當用戶端向伺服器端提出請求，則使用者對網頁的操作或互動就因此而被中斷了。因此，在此篇論文會在用戶端中使用 Ajax 的非同步溝通的方式，就是為了讓使用者能在編輯的時候，不需要將編輯的狀態中斷，也可以從伺服器端那邊獲得是否有最新版本的資訊。

2. 研究目的

在本研究中主要的目的，是希望能夠藉由將現有相關的演算法並結合 Ajax 技術的優點，提供使用者能在 Wiki 的共同創作環境下，當多個使用者同時編輯同一個條目時，若有產生編輯衝突的問題時，能夠立即讓正在編輯的使用者得知在伺服器端已經有較新的版本內容，使得編輯中的使用者能夠提早做進一步的整合。

3. 系統架構與實作

在接下來的章節中，在 3.1 節會介紹我們所創建的系統環境，3.2 節是說明整個系統的執行流程，3.3 節則針對 Diff3 演算法中的共同原始版本選擇來做討論，3.4 節介紹系統實作範例。

3.1 系統環境

在此篇論文中並沒有創建整個 Wiki 環境，由於此篇論文是單獨針對在編輯的部份來做探討，所以我們簡單的建立一個編輯環境。網站伺服器及資料庫系統分別為 Apache 以及 MySQL 資料庫，伺服器端採用 PHP 作為程式設計語言，在用戶端則使用 JavaScript 結合 Ajax 技術作為程式設計語言。

3.2 系統執行程序

本節所使用的版本符號定義如下：

V	用戶端使用者 U 所編輯的原始版本編號
Vx	用戶端儲存由伺服器端所傳來目前在伺服器中的最新版本編號，初值為 V
Vs	目前伺服器端所持有的最新版本編號
Vs’	Vs 所使用的原始版本編號

當在用戶端的使用者 U 對目前所看到的內容不滿意時，想要編輯，則進入編輯的環境中，此時用戶端會收到伺服器端所傳來的最新版本 V 的內容，同時用戶端會紀錄伺服器端目前最新版本編號 Vx 為 V。

使用者 U 開始編輯後，其所在的用戶端會藉由 Ajax 的技術，週期性的（例如：每隔 2 秒）將 V 與 Vx 版本編號傳送到伺服器端，再由伺服器端這邊接收到的 Vx 編號與目前伺服器端所存放的最新版本 Vs 編號做比較。若 Vs 與 Vx 編號相等時，表示使用者 U 所記錄的最新版本編號 Vx 與伺服器端所存放的最新版本 Vs 是相同的，因此伺服器端不需做任何回應動作；否則，當版本 Vx < Vs 時，表示伺服器端已產生了更新的版本 Vs，亦即在使用者 U 編輯的過程中，已有其他使用者編輯完畢存檔，所以此時有編輯衝突的問題產生；接著伺服器端則根據 Vs 的原始版本 Vs’與使用者 U 所編輯的原始版本 V 做比較，若 Vs’< V，則將 Vs 版本內容與使用者 U 目前所編輯而成的暫時內容的共同原始版本設定為 Vs’，此時伺服器端必須將版本 Vs 與 Vs’的內容一起回傳給用戶端，否則（Vs’>= V）將 Vs 版本內容與使用者 U 目前所編輯而成的暫時內容的共同版本設定為 V，此時伺服器端只需將版本 Vs 的內容回傳給用戶端(共同版本 V 的內容已存在用戶端中，所以不用傳送)。

接著用戶端根據(1)使用者 U 依據版本 V 所編輯而成的暫時內容 X，(2)從伺服器端所收到的最新版本 Vs 內容與(3)X 與 Vs 內容的共同原始版本內容（可能是 Vs’或 V 版本內容），依照 Diff3 演算法來做內容比較與合併的動作；此時用戶端並且更新伺服器端目前最新版本編號 Vx 為 Vs。

合併之後的內容，則由使用者檢視是否符合自己的需求，若是，則依照此合併版本繼續做編輯；否則，放棄此合併內容，依照原先使用者編輯的內容，繼續做編輯的動作。但不論合併與否，由於我們已在用戶端更新了伺服器目前最新的本版 Vx 為 Vs，所以當用戶端再一次以 Ajax 向伺服器端傳送

檢查是否有新的版本出現時，伺服器端並不會再一次地執行上述版本比較與傳送版本內容至用戶端的動作，除非真的有另一個更新的版本產生，才會再度執行上述動作。

圖 3 所示，此即為系統的執行流程圖：

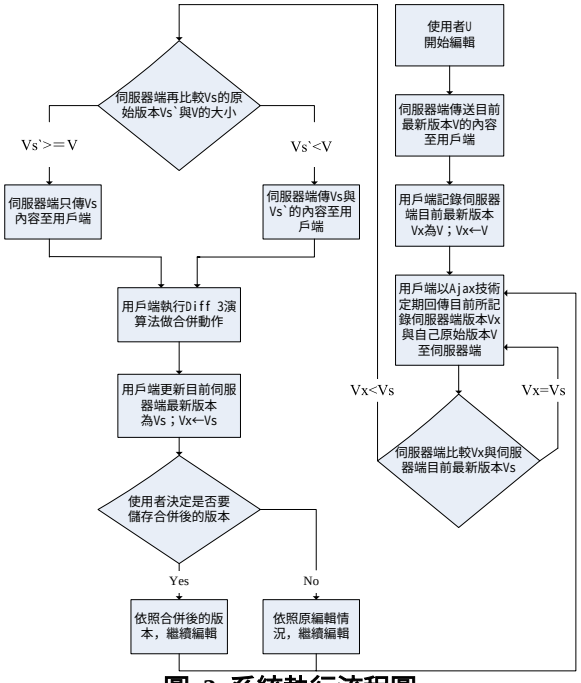

圖 3 系統執行流程圖

3.3 Diff3 演算法中所用的共同原始版本

如前言所述當 Diff3 演算法要做檔案合併的時候，必須有三個檔案 O、A、B，而 A、B 都是根據同一個原始版本 O 編輯而來的。因此當 A、B 都是依照原始版本做編輯時，我們要做合併的動作時，不會發生有原始版本不同的問題，如圖 4 所示，使用者 1 與使用者 2 是根據同一個原始版本 V 來做編輯的，在時間 t3 時使用者 2 執行存檔的動作產生 V2 版本，此時伺服器端會告知使用者 1 有新的版本產生，並且在使用者 1 的用戶端會執行 Diff3 演算法，進行檔案合併的動作，由於使用者 1 與使用者 2 所編輯的原始版本都是來自版本 V，所以並不會發生原始版本不同的問題；不過在時間 t4 時若有使用者 3 想要編輯，則使用者 3 所接收到的原始版本為 V2，而在使用者 1 仍在編輯的時間內，使用者 3 在時間 t5 做儲存的動作而產生 V3 版本，此時使用者 1 又會接收通知有新的版本儲存，並且執行 Diff3 演算法來做合併的動作，但此時問題產生了，由 Diff3 演算法的理論可知，原始版本必須是相同的，而現在使用者 1 與使用者 3 的原始版本分別為 V、V2，那我們該如何從中選擇一個版本來當作原始版本？在上述的例子中，因為 V 與 V2 版本的差異就在於使用者 2 所做過的編輯部分，我們可以先假設使用者 2 並沒有做此編輯，接著使用者 3 編輯時，他的部份想法與使用者 2 是相同的，即使用者 3 先作了與使用者 2 相同的編輯動作，接著使用者 3 又作了其他後續的修改，所以我們可以把使用者 3 的原始版本 V2 往前延伸到使用者 2 的原始版本 V，由 V 當作使用者 3 的原始版本，而忽略使用者 2 曾經編輯完成版本 V2 的過程。因此在這裡所考慮到的問題：原始版本不同時，對於使用者 1 要做合併動作時，原始版本 V 與 V2 不同的問題，我們則從中選擇最舊的版本（此例為 V）當作原始版本（參考圖 3 中 Vs'與 V 的比較部分），如此則可避免原始版本不同的問題。

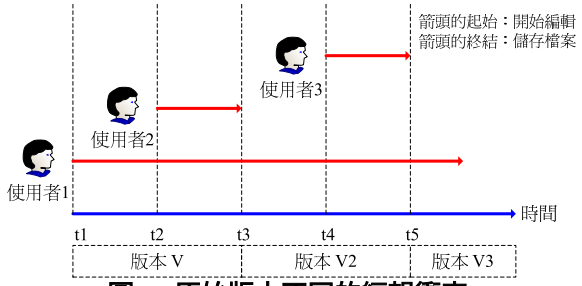

圖 4 原始版本不同的編輯衝突

3.4 範例說明

以下所呈現的是一個在較簡單的環境中所做的編輯測試：

最初所呈現給使用者的內容，如圖 5 所示。

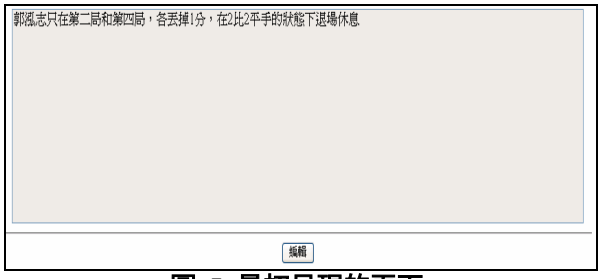

圖 5 最初呈現的頁面

而若使用者 1 想要編輯，則按下「編輯」的按鈕，來對內容做編輯，如圖 6 所示。

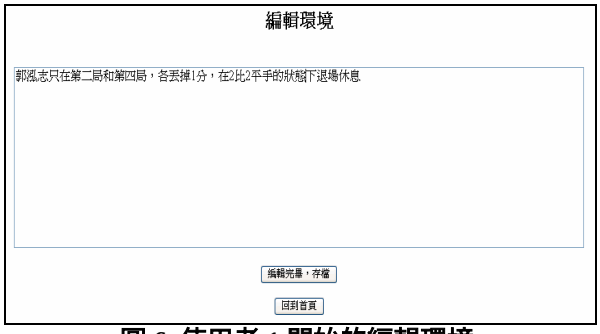

圖 6 使用者 1 開始的編輯環境

在使用者 1 編輯的過程中，如果有使用者 2 也想要編輯，則使用者 2 進入編輯環境開始編輯，其所編輯完成的內容如圖 7 所示。而在使用者 1 還未

編輯完畢，而使用者 2 已經完成他的編輯，並且作儲存的動作。假設此時使用者 1 編輯的內容如圖 8 所示，並且會收到一個訊息通知，告知使用者目前伺服器端有新的版本。

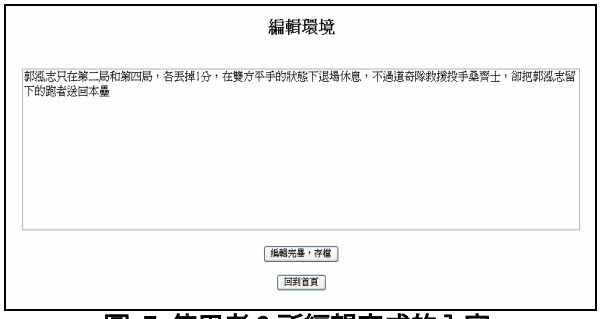

圖 7 使用者 2 所編輯完成的內容

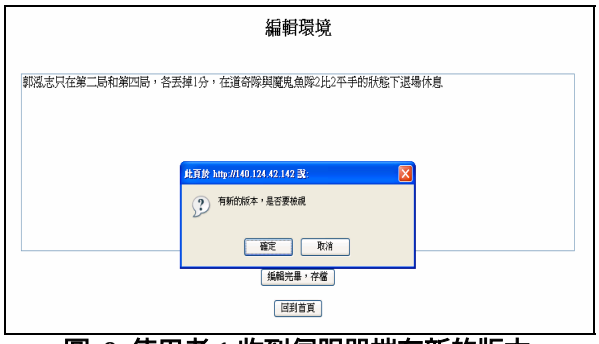

圖 8 使用者 1 收到伺服器端有新的版本

而使用者 1 接收到通知之後，可以依照個人的意願決定使否要檢視現在伺服器端中所存的最新版本，按「取消」，則回到原先的編輯環境繼續編輯；如果按「確定」則接著下一步驟，在此我們將原始版本（圖 5 內容）、使用者 1 編輯中的內容（圖 8 內容）以及伺服器端所傳的最新版本內容（圖 7 內容）都呈現給使用者觀看，如圖 9 所示。

圖 9 使用者 1 用戶端中呈現原始版本、使用者 1 編輯中的內容、伺服器端所傳的最新版本內容

此時使用者按下合併後的內容，就會將透過使用者 1 用戶端的 Diff3 演算法執行合併後的結果呈現給使用者 1 看，如圖 10 所示。

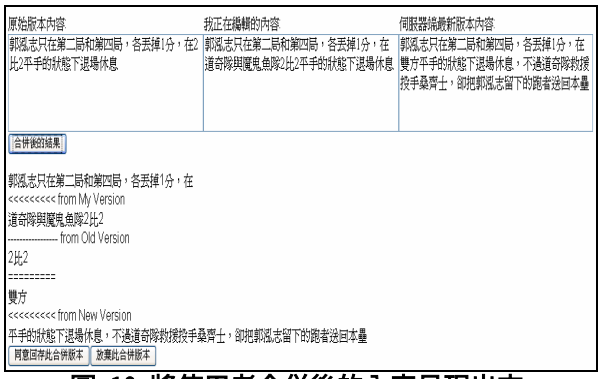

圖 10 將使用者合併後的內容呈現出來

此時使用者 1 看到合併後的內容，可以選擇是否要保有這些合併的版本，若「放棄此合併版本」，則會回到使用者原先編輯的內容，而忽視目前合併的結果；若「同意回存此合併版本」，就會如圖 11 所示，會將使用者在圖 10 所看到合併後的結果複製到使用者編輯環境中，讓使用者繼續編輯。

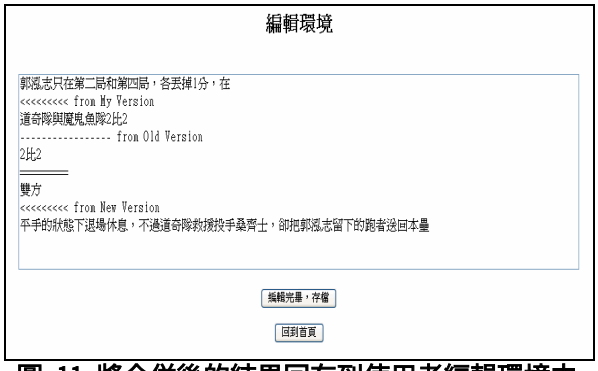

圖 11 將合併後的結果回存到使用者編輯環境中

4. 與現有技術的比較

在本節主要是將現有技術與本研究中所採用的方法做一個比較結果，如下表所示。

表 1 與「贏家能得到全部」做比較

贏家能得到全部	本論文採用的方法
使用者儲存時，忽略之前新的版本，直接覆蓋	使用者儲存時，可直接覆蓋新的版本，而其他使用者透過 Ajax 技術來得知伺服器端有新的版本
在使用者編輯途中，中間有新增幾個版本，使用者並不會知道	在使用者編輯時，若有其他使用者儲存新的版本，使用者可以立即知道，並做進一步合併的動作
每一次有新的版本被儲存都不知道	可以立即知道每一次新的版本被儲存
可以同時多人對同一條目編輯，但沒有考慮與其他人原始版本不同的問題	可以同時多人對同一條目編輯，且考慮與其他人原始版本不同的問題

表 2 與「不允許直接覆蓋新的版本」做比較

不允許直接覆蓋新的版本	本論文採用的方法
能同時多人對同一條目做編輯	能同時多人對同一條目做編輯
當使用者在編輯時，若有新版本儲存，則使用者無法得知，只有在使用者儲存時，才知道有新版本	在使用者編輯時，若有其他使用者儲存新的版本，使用者可以立即知道，並做進一步合併的動作
每一次有新的版本被儲存都不知道	可以立即知道每一次新的版本被儲存
使用者在儲存時，若之前有新的版本先儲存，則不能直接覆蓋，需要靠使用者手動的做修改	使用者儲存時，可直接覆蓋新的版本，而其他使用者透過 Ajax 技術來得知伺服器端有新的版本

表 3 與「鎖定頁面」做比較

鎖定頁面	本論文採用的方法
一個時間內，只有一個使用者可以對一個條目做編輯	可以多人同時對同一條目做編輯
需要考慮使用者編輯時間的 time-out 要設定多久	不需考慮編輯時間的長短
在使用者編輯時，不會有新版本的產生	使用者編輯時，可能會有多個新版本的出現
需考慮同一個使用者可以對同一條目接連編輯幾次	使用者可以持續編輯，直到作儲存的動作

表 4 與「使用自動合併的方法」做比較

使用自動合併的方法	本論文採用的方法
可以多人同時對同一條目做編輯	可以多人同時對同一條目做編輯
伺服器端執行 Diff3 演算法的動作，若同時有許多人同時對同一條目做編輯，此舉會增加伺服器的負擔	由用戶端執行 Diff3 演算法的動作，若同時有許多人同時對同一條目做編輯，則是由各自的用戶端分擔此一工作，避免伺服器負擔過重
當使用者在編輯時，若有新版本儲存，則使用者無法立即得知，只有在當使用者儲存條目時，才知道有新版本	在使用者編輯時，若有其他使用者儲存新的版本，使用者可以立即知道，並做進一步合併的動作

5. 結論

此篇論文說明了在 Wiki 開放式共同創作環境中，對於使用者而言，在編輯創作時，很可能會因為有其他使用者編輯的關係而產生編輯衝突的問題，在目前 Wiki 環境中，所提出的解決方法有，「贏家能得到全部」、「鎖定頁面」、「使用自動合併」、「不允許覆蓋新的版本」數種方法，但是實際應用上對於編輯衝突，仍不是這麼有效的解決此問題。

因此，在此篇論文中提出了使用 Diff3 演算及 Ajax 技術，以求更有效的解決問題。此研究選擇在用戶端使用 Diff3 演算法，其原因就是在以前的 Diff3 演算法都是建置在伺服器端，但是由於網路世界不分國界，使用者眾多，若是同時有多人使用到 Diff3 工具時，則會造成伺服器端的負擔過重，因此將 Diff3 演算法建置在用戶端，減輕伺服器端的負擔，就是我們考量之一；以往使用者在使用網頁時，用戶端要與伺服器端溝通，必須等待使用者送出請求才可以，但此舉會造成使用者對網頁的操作中斷，使用者沒有辦法一邊瀏覽編輯網頁，一邊主動向伺服器端連繫，如此對於 Wiki 編輯環境而言，使用者只能在編輯完畢並且作儲存的動作時，才能從伺服器端得知是否最新版本已經被變更了，這對於一個使用者編輯是相當不便的，因此在此論文中使用了 Ajax 非同步溝通的技術，藉此技術使得使用者在編輯的過程中，若伺服器端有新的版本回存，也能馬上得到有新版本回存的訊息，再經由使用者決定如何做進一步的編輯，讓使用者能在編輯的過程能夠更便利、簡單、即時。

參考文獻

[1] 呂雪惠，“新一代以語意為基礎的 Wiki 系統”，國立台灣科技大學資訊工程系碩士論文 2004

[2] 林信成、陳瑩潔，“Wiki 協作系統在數位典藏內容加值之應用研究”，TANet2005 研討會論文集

[3] 張雅惠，“多使用者編輯版本之資料庫維護與管理-以道路資料為例”，國立成功大學測量工程學系碩士論文 2003

線上資源

[4] Wiki History
http://c2.com/cgi-bin/wiki?WikiHistory

[5] WikiPedia
http://www.wikipedia.org/

[6] WikiWikiWeb
http://c2.com/cgi-bin/wiki?WikiWikiWeb

[7] Meatball Wiki, “EditConflict”
http://www.usemod.com/cgi-bin/mb.pl?EditConflict

[8] MoinMoin
http://moinmoin.wikiwikiweb.de/TourBusStop

[9] TWiki
http://twiki.org/

[10] WikiPedia, “Edit Conflict”
http://wikipedia.org/wiki/Wikipedia:Edit_conflict

[11] WikkiTikkiTavi
http://tavi.sourceforge.net/

[12] Oddmuse
http://www.oddmuse.org/

[13] WikiPedia, “Ajax”
http://wikipedia.org/wiki/Ajax_(programming)

[14] Sanjeev Khanna, Keshav Kunal, and Benjamin C.

Pierce, “A Formal Investigation of Diff3”, http://www.cis.upenn.edu/~bcpierce/papers/diff3-short.pdf

[15]Per Cederqvist, “Version Management with CVS” http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.9/cederqvist-1.12.9.pdf

[16] WikiPedia, “Longest Common Subsequence” http://wikipedia.org/wiki/Longest_common_subsequence

[17] WikiPedia, “JavaScript” http://wikipedia.org/wiki/Javascript