P2P網路去污染檔案容錯機制之研究

Fault-Tolerant Mechanism for Removing Polluted Files in Peer-to-Peer Networks

洪人傑　　王宗一　　鍾哲民

成功大學工程科學系

akolu0@yahoo.com.tw　　wti535@mail.ncku.edu.tw　　Adam145love@yahoo.com.tw

摘要

近年來世界各國不留餘力抵制 P2P 網路上的侵權行為，為此可以說是不惜採用激烈手段。但不可否認的，以 P2P 網路為主的檔案分享系統依舊是極具效率的，若此後 P2P 網路往分享授權檔案和使用者付費的方面繼續前進，相信 P2P 分享系統為最佳選擇。P2P 網路上的檔案汙染問題之造成是針對 P2P 網路分享侵權軟體所做的反擊，但若將來 P2P 網路上所分享之檔案都得到合法授權後，防制汙染就為一門重要的課題。

受汙染檔案(Polluted Files)，是指內容與標題不符的檔案或是檔案某部分為損壞或滲入不屬於此檔案的資料。P2P 網路中的檔案安全性是相當低的，除非已下載此檔案否則將無法辨別此檔案的正確性。

本研究使用 RAID 上的容錯機制來增加檔案的安全性，使其被汙染時可自行復原原始資料。此容錯機制可有效地抑制汙染，使一個受攻擊中 P2P 網路正確檔案擷取率上升，並在短時間中將 P2P 網路分享環境復原。

關鍵詞：檔案汙染、P2P、EVENODD

Abstract

Nowadays, there are too many illegal files in any kind of Peer-to-Peer file sharing systems and many governments try to stop the illegal behaviors. But it is unstoppable to avoid using the P2P file sharing system, because sharing files in the p2p network is the most efficient way to spread a file around. After the decriminalization of P2P file sharing. It must be a trend that if you want to share files, they must be authorized files. Thus, it is an important problem about polluted files.

How to propose an anti-pollution mechanism in the P2P network is important in the future. Content pollution is the most common form of pollution currently. P2P networks are vulnerable to pollution attacks, and files have no way to protect themselves. We cannot recognize if a file is a decoy unless you download it. There are less ways to make sure your downloading file is correct. When we got a polluted file, we just search another file and restart the download in most time.

This thesis proposes a fault tolerant mechanism to improve the safety of a sharing file. It can efficiently decrease files that are polluted and recover the environment of the P2P network from attacked in a short time.

Keyword: File Pollution、P2P、EVENODD

一、前言

P2P 檔案分享系統(Peer-to-Peer File Sharing System)，在近幾年不斷引起爭議，因為在此系統中可快速地分享各種的檔案，卻也充斥著大量不合法檔案的分享以及侵權行為[9]。但不可否認的，以 P2P 網路為主的檔案分享系統是極具效率的。P2P 網路分享系統的汙染問題於西元 2005 年 Infocom 會議上由 Liang 等人提出[6]。

使用 P2P 軟體來分享檔案最大的優點在於資訊的快速流通，但此一優點卻造成盜版軟體及病毒等[11]在網路上也快速地散播，當 P2P 網路遭受汙染攻擊時，也是同樣地擴張迅速[1, 3]。由於其網路平台上所

分享的資源大多都為電影、音樂等影音檔案，使得影音產業獲利大幅衰退，各國創作者也為了反盜版而發起各種自救活動。在這樣的環境下，導致一種新型態的公司的誕生，如 Overpeer 和 MediaDefender，專門接受影音產業和遊戲產業的委託，以各種方法攻擊和汙染 P2P 網路，讓檔案的取得率下降，造成使用者對此網路環境的不信任感，進而離開此 P2P 網路。以 KaZaA[8]為例，KaZaA 是一個以分享 MP3 為主的 P2P 分享軟體，擁有超越三百萬的使用者，超過五千 TB 的分享檔案，是美國最受歡迎的 P2P 分享軟體之一，卻在 2005 年時也因蓄意的污染攻擊，造成有 50%~80%分享檔案受到汙染[6]。而其他 P2P 分享軟體，如 eDonkey 也有 50%左右熱門的檔案遭受汙染[7]。

KaZaA因其平台上有許多未授權檔案，近幾年來遭受許許多多影音業者控告。美國唱片協會(RIAA，Recording Industry Association of America)亦在西元2003年，對沙曼網路公司（Sharman Networks，KaZaA銷售商）提出了多項告訴。此案已在西元2006年7月26日結審，沙曼公司除了賠償一億美金之外，更同意將KaZaA轉變為提供合法的音樂下載服務。而許多P2P軟體也有一樣的情形，如台灣的ezPeer，相信未來的會有更多P2P軟體加入合法化的行列，而問題將會從如何抑制P2P網路分享侵權軟體轉變為如何保護P2P網路上所分享的檔案，使得消費者能有效地下載所需的檔案[10]。根據惡意攻擊者所採取的策略，可以大約分類成3種[6, 7]。

(1)以檔案內容為主的汙染行為:

由攻擊者製造許多誘餌來使得良性的使用者所下載之檔案遭受汙染，而誘餌主要是修改一個特定檔案之內容且和擁有此特定檔案相同的敘述檔。將原本特定檔案之內容加入一些無法解碼的雜訊或者是將內容替換成不同檔案的內容。由於無法在下載檔案前，分辨是否檔案損壞或受到汙染，使得此方法成為一個相當簡單且有效的污染方式，也是本研究針對的問題。

(2)以敘述檔為主的汙染行為:

就如同字面上的意義，此種汙染方法主要針對敘述檔(metadata)之內容。針對想汙染之特定檔案，將不同的檔案的敘述檔內容，如檔名和檔案類型等，都改為與欲汙染檔案之敘述檔一致。以此來使得使用者選擇一個完全不相同的檔案。

(3)以索引為主的汙染行為:

此方法主要是針對有搜尋檔案能力的P2P網路架構，如以分散性雜湊表為基礎的P2P網路。攻擊者加入某一P2P網路後，將假的檔案紀錄回傳負責記錄的點，如檔案分享者IP、Port和檔案雜湊後的值等，或者修改本身所存放的檔案紀錄表格，來讓網路上看的到許多檔案分享，但卻無實體檔案之存在。

在下面的章節，會詳細敘述如何使用 EVEN-ODD 編碼方式來達到反制蓄意汙染檔案的行為。首先描述一個污染的環境是如何產生的，並且用一個數學模型來呈現，以及分析所帶來的影響，此模型的部分假設採用和Fluid Modeling[5]相同的設定。接著在檔案的分享上，使用一個可容錯編碼的方式。在檔案分割前先經過一連串編碼，接著分析此方法在一個受汙染的環境所帶來的影響。

二、系統架構

在一個點對點(peer-to-peer)的檔案分享環境下，每一個節點(peer)都是一位檔案提供者兼下載客戶端，同時分享和下載檔案。首先第一步簡化我們的問題，一份遭受污染的檔案一定會有一位以上檔案提供者,他們所使用的節點(peer)稱做攻擊節點(attacking peer)；其它非攻擊節點，則稱為良性節點(benign peer)，所以可將整個 P2P 分享網路上的節點分成這 2 大部分，如圖 1 所示。就目前的 P2P 分享軟體而言，良性節點並無法判斷哪個節點是攻擊節點，因此就算所下載的片段是受汙染的片段，也無法判定所要求提供片段的節點是為攻擊節點，而這也是造成汙染擴散的主要原因

之一。

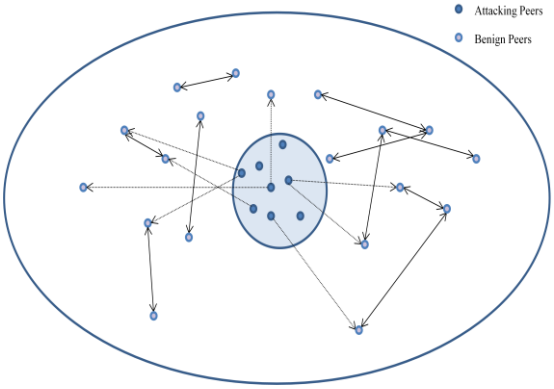

圖 1　P2P 網路中的節點行為

首先來說明一些名詞，在一個 P2P 網路上分享的主流多半為歌曲、電影、文件檔(如小說、論文等)和軟體等，這些分享的內容一定都會有一個「標題」(title)，而同一個標題會有著許多不同「版本」(version)，如古典音樂，同一個鋼琴練習曲，就會有好幾個演奏家的版本。當這些版本在 P2P 網路上流動時，取得此版本的節點所擁有的檔案稱為此版本的「副本」(copy)。

在 P2P 網路中，由於允許每一個節點可以自由的加入退出、分享檔案、停止分享等，使得使用者的行為變得太過自由、複雜，對於分析 P2P 網路帶來很大的難度，為使後面的系統分析變得有條理且易懂，針對此研究系統，以下將對使用者行為做一個合理的假設:

1. 當節點得到一個完整良好的副本後，節點不會離開此網路，而繼續分享此一檔案。
2. 若發現所下載的副本遭受汙染，立刻重新開始搜尋並下載檔案。

依循上面的假設，此系統 P2P 網路裡的每一個節點都只會有 3 種狀態:

1. 有著良好的副本的節點。
2. 有著受汙染副本的節點。
3. 沒有副本的節點。

而且每一個節點只會擁有一份副本。

現在使用 Markov process 來看此系統的各個節點的狀態。如圖 2 所示，X 表示在這個 P2P 網路中擁有良好副本的節點數量；Y 為在這網路中擁有壞的副本的節點數。總共可以分成 4 個狀態:

1. (X,Y): 某一個節點下載到受汙染檔案後，又一次下載到汙染的副本。
2. (X,Y+1): 某一個節點在第一次下載得到一份受汙染的副本。
3. (X+1,Y): 某一個節點在第一次下載檔案就得到一份好的副本。
4. (X+1,Y-1): 某一個節點在曾經得到汙染副本後，下載成功良好的副本。

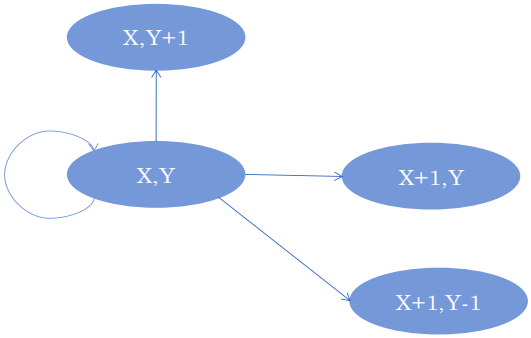

圖 2 此系統的 Markov process

三、模型推導

在推導數學模型前，先簡單定義系統環境的變數，假設在一個 P2P 分享網路上，有總數 M 個良性節點彼此分享檔案，而且就只有分享一個版本(version)的檔案，且此檔案被系統分割成 α 份片段(piece)(通常 P2P 分享軟體都將檔案分成數塊後才在網路上流動)。在這系統中有 x 份完整不受汙染的副本數(good copy)，x 隨著時間變動，而當一個 P2P 分享網路受到攻擊時，假定攻擊者會使用 N 個攻擊節點，且 N 為一個

常數。令受到汙染的副本數(polluted copy)有 y 份。如此可得知在時間 t 時，一個要求這份檔案的節點，選取到攻擊節點而得到一份受汙染檔案片段的機率為:

$$P(t) = \frac{y(t)+N}{x(t)+y(t)+N}$$

假定使用者發覺下載檔案遭受汙染是在開始下載後某一段時間，而這段時間假設為完成下載所需花費的時間。令所有使用者的平均發現時間為 1/μ，由此可知 μ 其實就為發現檔案遭受汙染的頻率。這樣可以得到在 x(t)在節點沒有擁有任何副本時，在時間 t 時，增加的速率為 $(M - x(t) - y(t))\mu(1 - P(t))^{\alpha}$。而一個擁有遭受汙染的副本節點在發現檔案受汙染再重新下載之後，得到一個良好的副本為 $y(t)\mu(1 - P(t))^{\alpha}$。總和以上的增加速率，可以發現:

$$\dot{x}(t) = (M - x(t) - y(t))\mu\big(1 - P(t)\big)^{\alpha} + y(t)\mu\big(1 - P(t)\big)^{\alpha}$$

同樣的在 y(t)的成長速率也可以得到類似的方程式，只不過由於將每一個副本都以 α 個片段來作分享，以使用者的觀點來看，只要檔案有某一片段受到汙染就為受汙染檔案(Polluted Copy)。必須將各片段受汙染的情況一併討論。所以，一個沒有擁有任何片段的節點從其他節點下載到一個以上污染片段的機率，就為一連串的排列組合:

$$P(t)^{\alpha} + C_1^{\alpha} * P(t)^{\alpha-1} * \big(1 - P(t)\big)^1 + C_2^{\alpha} * P(t)^{\alpha-2} * \big(1 - P(t)\big)^2 + \cdots + C_1^{\alpha} * P(t)^1 * \big(1 - P(t)\big)^{\alpha-1}$$

為考慮全部片段受汙染、一個片段受汙染、兩個片段受汙染等，直到所下載的片段全都不受汙染為止的組合。而節點曾經下載受汙染片段後，重新開始下載動作而得到一份良好完整的副本，為 $y(t)\mu(1 - P(t))^{\alpha}$ 。綜合以上的結果，可以得到$\dot{y}(t)$為:

$$\begin{aligned}(M - x(t) - y(t))\mu\{&P(t)^{\alpha} + C_1^{\alpha} * P(t)^{\alpha-1} * (1 - p(t))^1 \\ &+ C_2^{\alpha} * P(t)^{\alpha-2} * (1 - p(t))^2 + \cdots + C_1^{\alpha} * P(t)^1 * (1 - p(t))^{\alpha}\} \\ &- y(t)\mu(1 - P(t))^{\alpha}\end{aligned}$$

由於無法確切得知 x(t)之解，所以利用數值分析的方式來逼近此方程式之解。令初始值 x(0) = 1000 ，N = 1000，α = 5，μ = 1 (query/day)，M = 100000，如圖 3 所示，會發現一個相當極端的現象。下載受汙染片段的節點幾乎無法在下一次的下載中，完成一個良好的副本。這樣的結果是可以預測的，由於一個節點可以得到一份完整良好的檔案的機率過於嚴苛，一旦 α 值越大，機率就越小，對於一個熱門的檔案來說，攻擊者如果可以在初期就建立和擁有良好副本的節點數相近的節點個數，是相當有威脅性的。

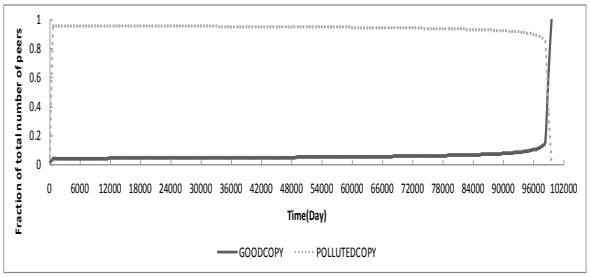

圖 3　M = 100000、x(0)/N = 1

四、使用 EVENODD 方法

為何在眾多的 RAID 的編碼方法中決定以 EVEN-ODD 來做為檔案的編碼?主要原因有兩個:

1. EVEN-ODD 編碼為對於兩個磁碟容錯之最佳編碼。

2. EVEN-ODD 編碼方式容易以軟體來實作。

此編碼應用於檔案分享前，先做分割的動作，然後以一定數量的片段做一個群組，以一個群組來做 EVEN-ODD 的編碼動作。如圖 4 所示，A1、A2 和 A3 為同一群

組，P1 和 P_d1 為 EVEN-ODD 編碼後所增加的資料(redundant data)。對於得到一個完整的檔案必須下載所有的片段而言，當此檔案是以 EVEN-ODD 方式編碼時，會使得檔案附加一些額外的片段，造成網路上的流量增加。如圖 5 所示，群組大小將影響檔案所造成的多餘負擔，最高達到原大小的 66%。但檔案經過以 EVEN-ODD 編碼過後，本身已經擁有一定程度容錯能力，如圖 6 所示。若檔案分成五個片段，以此五個片段作為一個群組，EVEN-ODD 編碼後會增加兩個片段，則此檔案會變成有七個片段。當某一節點下載了其中五個片段，只要這五個片段為良好無損壞的片段，此節點可自行以解碼的方式，產生尚未下載的兩個片段。因此使用 EVEN-ODD 編碼，可以減輕網路流量方面的多餘負擔。

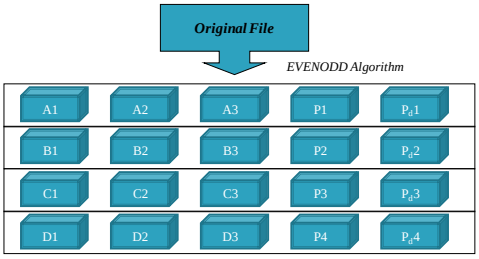

圖 3　EVEN-ODD 編碼

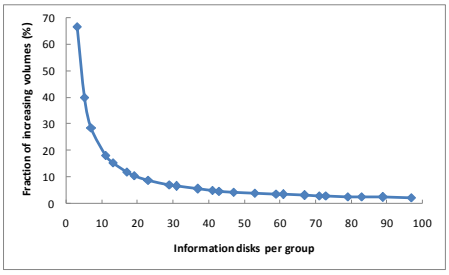

圖 4　EVEN-ODD 編碼群組的大小造成的負擔

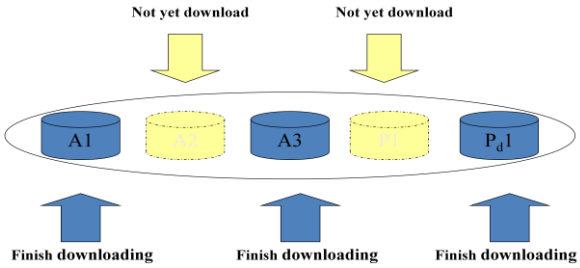

圖 5　可做復原的動作來恢復檔案

在容錯度方面，EVEN-ODD 編碼方法為在同一群組中，可容錯兩個錯誤片段。常見的 P2P 分享檔案之片段，大小為 512KB 至 2MB，以一部大約 1GB 大小的影片而言，檔案區塊大約 1000 至 2000 個，若將全體區塊做為同一群組，其容錯率大約 0.1%，幾乎無容錯能力可言，圖 7 為每個群組大小可以容忍的錯誤率，每一組群組中只容許兩個錯誤。為了增加整體檔案容錯率，只有將檔案區塊分成更多小群組，而所需付出的就是磁碟的空間。當然無論如何分群，以 EVENODD 編碼容錯率最大為 40%，無法跨過此限制。

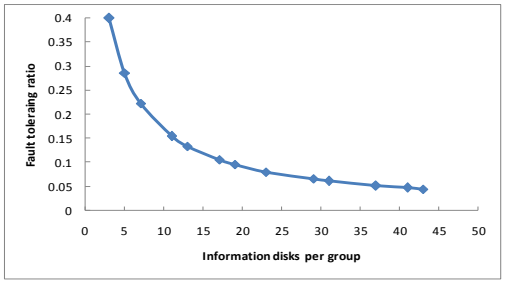

圖 6　群組容錯率

接下來以與前一章節相同的假設，來重新推導數學模型。由於使用 EVEN-ODD 編碼會使原本只有 α 個片段的檔案增加一些片段，為了不使模型複雜度太高，將以 α 個片段為一個群組，所以檔案會比原本的多出兩個片段。

首先來看 y(t)成長的方程式。同樣地，一個沒有擁有任何片段的節點下載到三個

以上污染片段的機率，也是一連串的排列組合，令此機率為 Py(t):

$$\begin{aligned}Py(t) = P(t)^{\alpha+2} &+ C_1^{\alpha+2} * P(t)^{\alpha+1} * \left(1 - P(t)\right)^1 \\ &+ C_2^{\alpha+2} * P(t)^{\alpha-2} * \left(1 - P(t)\right)^2 \\ &+ \cdots \\ &+ C_{\alpha-1}^{\alpha+2} * P(t)^3 * \left(1 - P(t)\right)^{\alpha-1}\end{aligned}$$

由於多了兩個片段所以受到汙染的機率應該比原本的來大，但因為檔案本身的還原能力，使得 Py(t)中下載一個和兩個受汙染檔案片段的機率消失。接著看節點曾經遭受汙染後，重新開始下載，得到一份良好完整的副本的機率 Px(t):

$$\begin{aligned}Px(t) = (1 - \left(P(t)\right)^{\alpha+2} &+ C_1^{\alpha+2} * \left(1 - P(t)\right)^{\alpha+1} \\ &* P(t) + C_2^{\alpha+2} * \left(1 - P(t)\right)^{\alpha} \\ &* P(t)^2\end{aligned}$$

因檔案有了復原的能力，在下載 α 個正確的片段後做解碼的工作，會使 Px(t)與 Py(t)總和大於 1,所以此設定在這邊不採用。不論是否已達到解碼的標準，也都必須下載所有的片段。上列方程式的第一項為重新開始下載後,得到的全部片段為良好的。後兩項為得到一個或兩個汙染片段但檔案解碼後可以得到完整正確的片段的機率。此兩項若未使用 EVEN-ODD 編碼,應被歸入 Py(t)中。

由前兩個機率，我們可得到ẏ(t)和ẋ(t)的方程式:

$$\dot{y}(t) = (M - x(t) - y(t))\mu Py(t) - y_t \mu Px(t)$$

$$\dot{x}(t) = (M - x(t) - y(t))\mu Px(t) + y_t \mu Px(t)$$

同樣地以上(7)(8)兩個方程式也為非線性方程式，而且更加複雜，而我們也無法利用這 2 個方程式來看出 x(t)和 y(t)的圖形曲線。如圖 8 所示，同樣是利用數值分析的方式來逼近，且初始值 x(0) = 1000 ，N = 1000，α = 5，μ = 1 (query/day)，M = 100000。與圖 3 的比較之下，可以明顯看出，受汙染檔案在攻擊者散播污染的版本後，可在短時間內使得受汙染的節點數量降低，且整個系統在第 16 天就收斂了，明顯達到反汙染的目的。這是使用一連串的假設而得出的結果。在下一個的章節，將會以模擬的方式來驗證是否正確。

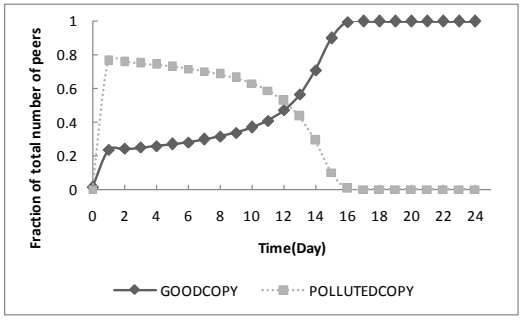

圖 7　M = 100000、x(0)/N = 1，with EVEN-ODD

五、系統模擬與結果

本研究目的為是否在分割檔案時使用 EVEN-ODD 編碼方式取得檔案同位元資料以增加此檔案的安全性，可使得一個受攻擊的 P2P 分享系統，不斷降低受汙染檔案之數目，並且有效的恢復受汙染的環境，使其保有原有檔案分享的效率。

（一）節點行為

一個 P2P 網路節點必須有下載檔案行為才會有遭受到汙染的可能性，因此實驗的重點會注重在節點本身的行為上。在一般 P2P 網路模擬上所注重的流量、架構以及搜尋時間等，皆不為此實驗中的考慮因素。本模擬對於一個 P2P 網路的假設如同第 3.1 章節及第 3.2 章節所設定。網路上初始分享完整正確檔案節點數、要求此檔案之良性節點數以及攻擊節點的數量在開始分享檔案時就為固定不變的，且所有節點不能離開此網路。完成檔案下載也一定要分享片段給其他未下載完成的節點，不能隨意停止分享動作。

每一個要求檔案的良性節點所要執行的工作都是相當單純的:

1. 選擇所缺少的檔案片段。
2. 選擇一個擁有此片段的節點作為提供片段給良性節點的對象。
3. 檢查是否已經下載結束，若已完成檔案下載則停止選擇片段及節點。

為了在每一次 P2P 網路上所有節點完成檔案下載時，能夠得知每一個節點所下載完畢的檔案是否為受汙染檔案，會設置一個監視的工作，來負責記錄節點是否受汙染。在一定時間後，所有節點都已完檔時，統計此 P2P 網路中所有節點的所下載的檔案屬性(完美或是受汙染)，並且將擁有受污染片段之節點重新開始下載動作。

（二）模擬結果

如圖 9 所示，根據上一節的節點工作設定，在不使用 EVEN-ODD 編碼檔案的 P2P 網路遭受攻擊時，令良性節點數為 2000、攻擊節點數為 100，以及分享良好檔案的節點數初始值也為 100，則起始時選擇攻擊節點和分享良好檔案之節點的機率是相等的。受汙染副本在此網路中經過 4 天後，大約百分之八十的節點遭受汙染。之後汙染副本所佔的比例開始大幅下降，在系統運作 10 天後，此網路所有的節點皆取得良好檔案之副本。

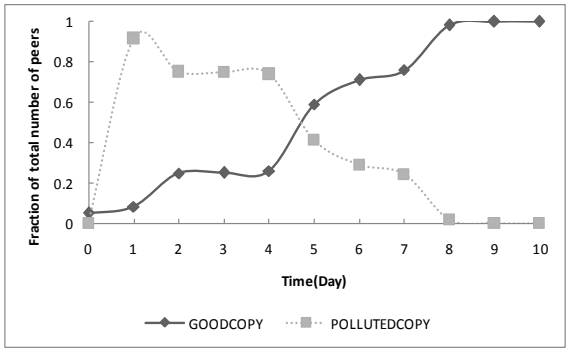

圖 9　P2P 網路檔案分布，不使用容錯機制

將此 P2P 網路加上檔案容錯機制後，如圖 10 所示，受汙染檔案的副本數，大約在一天後，就開始大幅下降。在此系統開始運作後的第四天，已使 P2P 網路中全部的良性節點獲得一份良好的檔案副本。與圖 9 比較可發現在加入檔案容錯機制後，受到汙染攻擊的時間下降，收斂時間比起不使用任何容錯機制的 P2P 網路快上 40%~60%的時間。

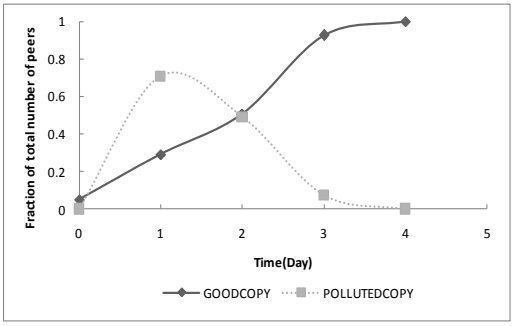

圖 10　使用 EVEN-ODD 編碼檔案後 P2P 網路檔案的分布情形

（三）問題討論

圖 11 和圖 12 的曲線為以第三章所推方程式之解，可以發現曲線的形狀相當接近，但 P2P 網路收斂的天數，卻比模擬的結果高出太多。將圖 10 與圖 12 比較，可以發現受汙染檔案副本數量，在第一天過後，兩者皆呈現很明顯的下降，唯一不同的是收斂的速度。由方程式得到的收斂速度大約需要 13 天，而模擬結果只花了 3 天。而圖 9 與圖 11 的差別更是明顯，且同樣地，在受汙染副本數量達到整體百分之七十左右曲線急速掉落。

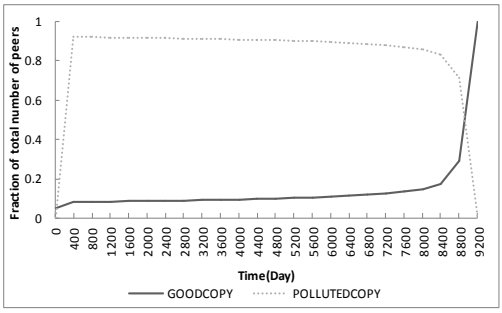

圖 11　方程式之解，當 M=2000、N=100，不使用容錯機制

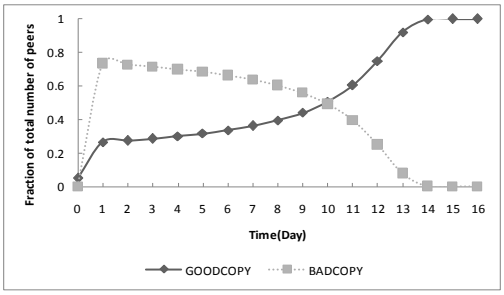

圖 12　方程式之解，當 M=2000、N=100，使用 EVEN-ODD 編碼方式

為了分析此一問題，另外記錄了在模擬環境中，每一個良好片段以及受汙染片段的數目，如表格 1 至表格 4 所示。可以發現檔案片段的數目都比得到一個完整良好副本之節點數目來的多。攻擊節點之數量以及分享良好檔案節點數量初始設定的比例為 1:1，所以在第一天的分享過程中，所有節點選取分享良好檔案之節點和攻擊節點的機率為相同。如表格 1 所示，在第一天結束後，受汙染片段以及良好片段的數目大致相同，打破此一平衡在於良性節點若抓取了一份完整的良好副本後，就不再變動，使得選取一份良好片段的機率得以累積，逐漸使得良好片段數目上升。而方程式的假設為每個節點從當下的 P2P 網路中選取片段，且因為使用數值方法的原因，時間為離散化，導致每個節點所見的 P2P 網路副本分佈為前一天的分佈情形。在選擇片段上，只要是從受汙染的節點中所獲取的片段一律為受汙染片段，並使得自己成為受汙染節點中的一員。因此每一個節點可選擇取得良好片段之節點數就同等於擁有完整良好副本之數目，所以可知在方程式中的節點累積良好檔案的速度會較模擬中的節點緩慢。此一情形當網路節點數越多，就越發明顯，如圖 3 所示。

表 1 每天良好片段的數量，不使用容錯機制

Day / Piece	1	2	3	4	5	6	7	8	9	10
1	1118	579	977	1674	2000	2000	1862	2000	2000	2000
2	1051	708	986	715	1033	1186	1740	1953	2000	2000
3	1121	1232	1208	1440	848	1034	1220	1314	1699	2000
4	1104	1259	1552	2000	1948	1893	2000	2000	2000	2000
5	1137	1214	1250	1935	1950	2000	2000	2000	2000	2000

表 2 每一天副本的分布數量，不使用容錯機制

Day / Copy	1	2	3	4	5	6	7	8	9	10
Good	199	288	406	555	696	821	960	1314	1699	2000
Polluted	1801	1712	1594	1445	1304	1179	1040	686	301	0

表 3 每一天良好片段的數量，使用 EVEN-ODD

Day / Piece	1	2	3	4
1	1200	969	1896	1991
2	1037	1250	1514	2000
3	871	1144	1954	2000
4	904	1157	1134	1999
5	1091	1291	1701	2000
6	892	1445	1793	2000
7	1063	1676	2000	1900

表 4 每一天副本的分布數量，使用 EVEN-ODD

Day / Copy	1	2	3	4
Good	458	839	1822	2000
Polluted	1542	1161	178	0

由於方程式將副本中每個片段的詳細情形忽略，以是否存在一個以上污染片段做為受汙染基準，也就是若一個片段受汙染則整個副本皆受汙染。所以判定某一副本為受汙染副本時，也將副本中的其他良好之片段直接判定為受汙染片段，此一假設使得受汙染之效應被放大。

在此修改一個假設，且此修改假設乃在使用者可能做出的行為中。原本的假設為下載完成後若是檔案遭受污染則直接重新開始下載，更改為重新下載前先將前次所下載的檔案片段全部刪除。在此假設下，由於方程式的解是由數值方法來逼近，在一段時間內(t，t+△t)，所看見的網路副本分佈是相同的，若此段時間相等於 $1/\mu$，就可以將取得受汙染片段之機率改為:

$$P(t) = \frac{N}{x(t) + N}$$

圖 13 以及圖 14 為修改假設後方程式與模擬結果之比較。圖 13 為不使用容錯機制下，受汙染 P2P 網路的副本分佈，可以看出曲線已經相當接近，這是由於方程式中良好片段被判定為受汙染片段所造成的效應降低。圖 14 所示為使用 EVEN-ODD 容錯方法的受汙染 P2P 網路副本分佈，雖然曲線已經比修改假設前接近了許多，但明顯地仍有一小段落差，模擬環境的結果明顯收斂的較慢，這是由於原本模擬中可由片段累積所造成的取得良好片段機率下降，但方程式可得到良好片段機率卻大幅上升，因此造成此落差。

由此可見，推導出的方程式並不準確，原因在於方程式中各個副本片段並沒有分別，而是用相同的機率去假設，且在方程式中沒有確切的假設所下載的片段為受汙染或是良好的，只以下載片段的來源來做區別，以上兩點都是造成誤差的原因。

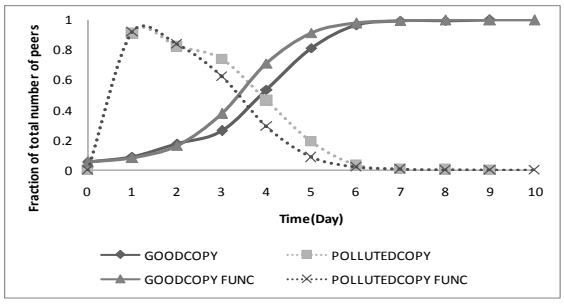

圖 13　修改假設後受汙染 P2P 網路環境副本分佈比較，不使用容錯機制

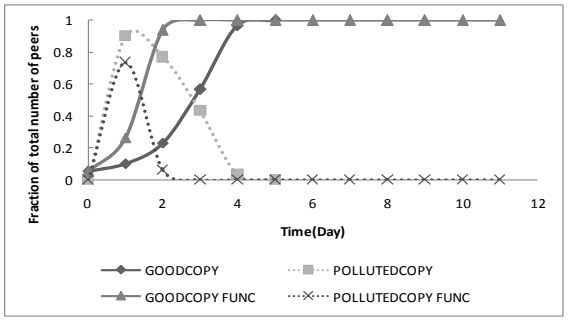

圖 14　修改假設後的受汙染 P2P 網路環境副本分佈比較，使用 EVEN-ODD 編碼方式

六、結論

在各國反盜版的聲浪下，各種 P2P 分享網路也將陸續限制使用者只能分享合法的檔案。如此一來 P2P 分享網路上的檔案將必須更加地需要保護，保證檔案本身內容的正確性。對於惡意的攻擊 P2P 網路的行為，未來不再是保護智慧財產權的一種手段，反而有可能將危害消費者的權益。本研究提出以 RAID 上的容錯編碼方式，可以使得一個受攻擊的 P2P 網路，有效地縮短 40~60%的受污染時間，降低遭受汙染

的節點數量，提供使用者一個乾淨的分享環境，且此方法可以容易地嵌入目前網路上的 P2P 網路分享軟體中，而不需去改變其 P2P 網路本身的架構。

七、未來展望

加強編碼的容錯能力：在本研究中，採用的 EVEN-ODD 編碼在一個群組中只能修復兩個以下的片段錯誤，若想使得檔案容錯能力更佳，只能將檔案分成更多的群組，這樣必須花費更多的儲存空間來放置一個檔案，且檔案容錯能力最高只達到容錯 40%的檔案錯誤(以 3 個片段為一個群組)。若使用更佳容錯能力的編碼，可有效的提升反汙染的效率，且降低電腦本身的負擔。

反制其他攻擊 P2P 網路的方法：本研究中的方法只能解決利用分享受汙染檔案的方式，造成使用者無法在短時間中得到所要的檔案。對於其他的攻擊方式:如利用空的檔案來汙染搜尋的結果(The index poison)[7] ，受到攻擊的 P2P 軟體主要為有搜尋檔案能力的 P2P 網路架構，如 eMule / **e**Donkey、KaZaA 等，或是阻斷服務攻擊[2, 4]癱瘓某些分享的節點等，皆無法處理。

八、參考文獻

[1] C. Cristiano, S. Vanessa, A. Jussara, and A. Virgilio, "Fighting pollution dissemination in peer-to-peer networks," in *Proceedings of the 2007 ACM symposium on Applied computing* Seoul, Korea: ACM Press, 2007.

[2] D.Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaaenepoel, "Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems," *ACM SIGMETRICS Performance Evaluation Review,* vol. 33, pp. 38-45, 2005.

[3] B. Fabricio, C. Cristiano, V. Marisa, A. Virgilio, A. Jussara, and M. Miranda, "Impact of peer incentives on the dissemination of polluted content," in *Proceedings of the 2006 ACM symposium on Applied computing* Dijon, France: ACM Press, 2006.

[4] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: An architecture for mitigating DDoS attacks," *Ieee Journal on Selected Areas in Communications,* vol. 22, pp. 176-188, Jan 2004.

[5] R. Kumar, D. D. Yao, A. Bagchi, K. W. Ross, and D. Rubenstein, "Fluid Modeling of Pollution Proliferation in P2P Networks," *ACM SIGMETRICS Performance Evaluation Review,* vol. 34, pp. 335-346, 2006.

[6] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P File Sharing Systems," *IEEE Infocom 2005,* March 2005.

[7] J. Liang, N. Naoumov, and K. W. Ross, "The Index Poisoning Attack in P2P File Sharing System," *IEEE Infocom 2006,* April 2006.

[8] S. G. Nathaniel and K. Aaron, "Usability and privacy: a study of Kazaa P2P file-sharing," in *Proceedings of the SIGCHI conference on Human factors in computing systems* Ft. Lauderdale, Florida, USA: ACM Press, 2003.

[9] C. Nicolas, S. W. Andreas, and C. John, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *Proceedings of the 6th ACM conference on Electronic commerce* Vancouver, BC, Canada: ACM Press, 2005.

[10] R. Pablo, T. See-Mong, and G. Christos, "On the feasibility of commercial, legal P2P content distribution," *SIGCOMM Comput. Commun. Rev.,* vol. 36, pp. 75-78, 2006.

[11] S. Seungwon, J. Jaeyeon, and B. Hari, "Malware prevalence in the KaZaA file-sharing network," in *Proceedings of the 6th ACM SIGCOMM on Internet measurement* Rio de Janeriro, Brazil: ACM Press, 2006.