在分波多工接取網路上的多重故障元件定位演算法之設計

A Design of Multi-failure Elements Localization Algorithm in WDM Access Networks

魏妙旭
高苑科技大學 通識教育中心
mswey@cc.kyu.edu.tw

吳介騫
高雄第一科技大學 電腦與通訊工程系
jcwu@ccms.nkfust.edu.tw

摘要

當分波多工網路發生故障時，便可能引發許多元件發出告警訊號，如何在蜂擁而至的告警訊號中定位出正確的故障元件，以盡速啟動網路回復機制是一件重要的課題。許多的文獻提出了網路多重故障定位演算法，然而，它們需花費大量的時間來建立一個龐大的資料庫。本文提出一個演算法，樹狀網路多重故障過濾定位演算法；它可節省建立資料庫之時間，但必須多花一些時間來過濾可能發生故障的元件。

關鍵詞：告警，多重故障，定位

Abstract

When failure occurs in WDM network, it may result in the large number of alarms. How to localize the position of failure components from these alarms and restore the networks quickly is important. Many methods have been proposed to position the failure components in the literatures. However they consume the enormous time to set up an huge database. In this paper, we propose an algorithm, namely Tree-topology-network Multi-failure Filtering Location Algorithm(TMFLA). It can reduce the time of setting up the database, but it will spend more time to filter the probable failure components, than the existing works.

Keywords : alarm，multi-failure，location

一、緒論

近年來由於網際網路的服務多元化，使得人們在網路上所傳輸的資料由過去的純文字型態轉變為包括了聲音、影像等的多媒體型態，人們對於網路頻寬的需求也就有爆炸性地增加，因此，如何快速地擴展網路頻寬是一項重要且迫切的議題。分波多工(Wavelength Division Multiplexing，WDM)技術是將光譜分成多個獨立的波長 (Wavelength)，使得不同波長的訊號能在同一條光纖上同時傳送，可以使原有的光纖網路倍增通訊的頻寬。所以，以分波多工技術為基礎的光纖網路，咸認為是現今提高網路頻寬的最佳的選擇之一。可是高容量的分波多工網路一旦發生故障時，網路斷訊將造成用戶大量資料的遺失並導致的極大損失。因此在網路故障時，如何在最短的時間內回復網路的運作，使損壞的影響減至最低，是一個重要的課題。[1]

但是，故障網路的回復機制只有在網路架構為網狀及環狀時才具有較高之存活力(Survivability)，因為網路結構可以在工作路徑之外建立保護路徑。而由於接取網路(Access Network)架構，大都屬於樹狀或星狀網路架構，其網路拓樸只有一條光纖連至光末端節點，因此，保護路徑只能建立在同一條光纖內，以另一蕊光纖作為備用路徑。但是一旦光纖如被挖斷，通常整條光纜內的光纖即全部斷裂，事先規劃的備用路徑亦無法使用，所以此類網路架構是較不具存活力的。因此，接取網路在事先規劃備用路徑既難以發揮作用，則如何在發生故障之後迅速定位出正確的故障元件位，以盡速進行搶

修，便成為急需解決的問題。

然而在分波多工網路中，只要有任一光元件故障，便可能引發許多元件發出告警 (alarm) 訊號，例如：因為某一光纖的斷裂，可能導致下游接收機因收不到信號而產生告警。而告警的多寡又與系統所使用的波長數有關，高密度分波多工網路(DWDM)的系統隨著使用波長數的增加而增加，當元件發生故障時，其所產生的告警也愈多，相對地定位出故障元件的困難度也愈高[6]。針對此問題，文獻[1]利用在樹狀網路中只有一個區段光送收系統，提出在每一光末端節點建立一條光徑的方法替代建立多路通道的觀念，來降低定位故障元件所需花費時間並提高定位故障元件的準確度，但其缺點是只能定位單一故障元件。

Carmen Mas 提出了告警過濾演算法(Alarm Filtering Algorithm，AFA)[2]，係利用光元件之特性、告警分類及領域(domain)之觀念，但由於樹狀網路拓樸架構上，由多個光被動元件所連續串接，使告警過濾演算法因此受到限制。

後來 Carmen Mas 又提出了故障定位演算法（Fault Location Algorithm，FLA）[3]，此演算法包含事先計算(pre-computation)階段及搜尋階段。其中，事先計算階段是為了先建構所有可能故障之二元樹(binary tree)的資料庫，其步驟為：(1)計算網路上每一個元件的領域；(2)將領域相同的元件歸為同一組，表示可能單一故障之候選元件集合；(3)依序將不同領域的兩組、三組、四組、、、n組元件做聯集，表示可能多重故障之候選元件集合；(4)將步驟(2)及(3)之處理結果，對應成二元向量(binary vector)，並對應到二元樹(深度為網路上告警元件之個數)上的一個葉子節點(leaf node)，每一葉子節點即為一種故障可能之候選元件集合。

當可能故障之二元樹的資料庫建立後，一旦收到元件所發出之告警，可將這些告警對應成二元向量，並從二元樹的根比對到相對應的葉子節點，即可找到故障之候選元件。此搜尋階段只需和資料庫的二元樹來比對，而不需任何的計算，縮短搜尋之時間，所以速度較快。

但故障定位演算法（FLA）必須事先建立一個耗費龐大記憶空間的資料庫，此資料庫之建立，也必須花費非常巨大的時間才能夠完成。所以它只適用於網路拓撲或通道長時間不改變的情形；但若對於網路拓撲或通道需經常性變更時的情形，卻要每次都重新計算資料庫，此作法浪費非常巨大的時間及空間，而不夠有效率性。

因此，本文的研究的目標是針對樹狀拓撲網路，提出一個可以改進 FLA 這項缺點的演算法，此演算法可以在接取網路架構上，快速又準確地過濾出可能發生的多重故障元件位置，且不需要浪費任何時間及空間來建立事先的資料庫。

本文其餘部份安排如下：第二節詳細描述我們所提出的演算法，並舉一範例以說明之；第三節是針對我們所提出的演算法和 FLA 演算法做模擬比較及效能分析；並在第四節做出結論。

二、樹狀網路多重故障過濾定位演算法(TMFLA)

本文所提出的樹狀網路多重故障過濾定位演算法(Tree-topology-network Multifailure Filtering Location Algorithm，TMFLA）是在接取網路架構上，利用光末端節點個數比光網路元件個數少的特性，及通道上之元件有否誘發光末端節點發出告警的觀念，來達到快速又準確地過濾出可能發生的多重故障元件位置之目標。

2.1 參數及符號之定義

我們定義在本論文中，演算法將使用到的一些符號、集合及運算如下：

e：光網路中之元件。

a：告警。

R：網管所收到的告警集合。

En：在網路中之光末端節點。

n_a：在網路中所有告警元件節點個數。

n_{na}：在網路中所有非告警元件節點個數。

n_c：在網路中之所有元件節點個數。

∨：point-wise OR 運算，例如
(1100)∨(1010)=(1110)。
∧：point-wise AND 之運算，例如
(1100)∧(1010)=(1000)。
~：point-wise NOT 之運算，例如
~(1100)=(0011)。

2.2 系統假設

TMFLA 演算法是特別針對接取網路為樹狀結構的多重故障元件定位演算法，樹狀網路的特性是只有一個區段光送收系統且為多個被動元件所串接而成的網路。首先我們做一些系統的假設：

(1) 告警均假設在理想狀況，不會有誤報警發生。
(2) 每一節點所送出的告警，可以經由獨立而可靠的其它網路來傳送給管理者，所以不會有遺失發生，管理者均能收到告警。
(3) 每條鏈結所使用的波長數沒有限制。
(4) 信號傳送方向，僅考慮由接取節點向光末端節點廣播之方向。
(5) 僅考慮由接取網路光末端節點所發出告警情形。
(6) 每一節點，發生故障之機率均相等。
(7) n 個元件同時故障之機率比 n+1 個元件同時故障之機率高。

2.3 樹狀網路多重故障過濾定位演算法

樹狀網路多重故障過濾定位演算法機制可分為下列四個模組，分別介紹如下：

(1) 上游建立(Source Establishing)：
接取網路連接骨幹網路之節點，稱為接取節點(access node)，由接取節點到每一光末端節點均建立一條通道，每一條通道所經過的元件為所有可能發生故障的元件，所以建立每一光末端節點的上游集合，並將其對應成二元向量。
即

$S_i = Source(En_i) = (e_{i1}, e_{i2}, \ldots, e_{ik})$

$\forall\ 1 \le i \le n_a, 1 \le i1, i2, \ldots, ik \le n_{na}$

且

$Bin(S_i) = \{(s_1, s_2, \ldots, s_{n_{na}}) \mid$ 若第 j 個元件節點故障而導致光末端節點 En_i 發出告警，則 s_j =1，否則 s_j = 0，$\forall\ 1 \le j \le n_{na}\}$。

(2) 告警收集(Alarm Collecting)：
網管收到告警後，將重複的告警捨棄, 且經過排序後，將其對應成二元向量。
即

$R = \{e_{i1}, e_{i2}, \ldots, e_{in}\}$，

$1 \le i1, i2, \ldots, in \le n_a$，

且

$Alarm = Bin(R) = \{(a_1, a_2, \ldots, a_{n_a}) \mid$ 若光末端節點 En_i 有發出告警，則 a_i =1，否則 a_i =0，$\forall\ 1 \le i \le n_a\}$。

(3) 元件過濾(Component Filtering)：
所有告警元件是否有發出告警，都有助於定位出可能發生故障之元件。根據有發出告警之元件，可以判斷出可能故障之元件，而根據沒有發出告警之元件，可以再過濾出正常之元件。
即

$CF_1 = \vee_{a_i=1} Bin(S_i)$，$1 \le i \le n_a$，

$CF_0 = \vee_{a_i=0} Bin(S_i)$，$1 \le i \le n_a$，

且

$CF = CF_1 \wedge (\sim CF_0)$。

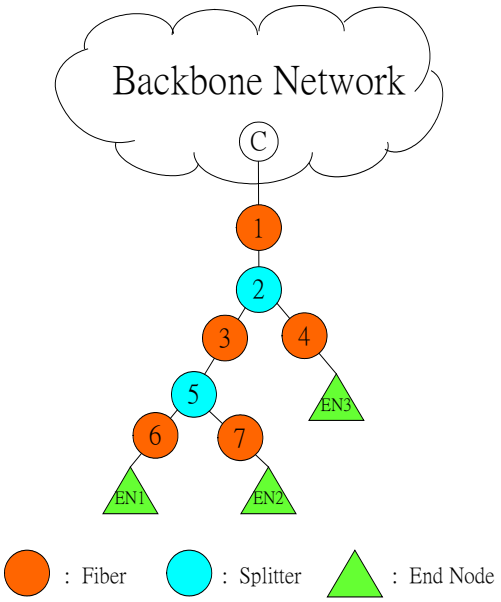
(4) 候選選擇(Candidate Selecting)：
因為在同一時間，n 個元件之故障比 n+1 個之故障的可能性高；所以，依據網路拓樸，選擇較上層之父節點元件取代較下層之兄弟節點元件，方式是依序由下層之兄弟節點，檢查是否同時故障，若有，則修改成為其父節點及祖父節點故障。
即

$CS = \{(s_1, s_2, \ldots, s_{n_{na}}) \mid \forall$ 兄弟節點 i、j，且 s_i、$s_j \in CF$，若 s_i =1 且 s_j =1，則 s_i =0、s_j =0，且節點 i、j 之父節點 k 及祖父節點 m 所對應之 s_k =1 及 s_m =1$\}$。

若結果 CS 中之 s_i=1，則表示元件節點 i 是故障點。

2.4 範例

茲舉一例說明 TMFLA 演算法：網路拓樸結構如圖(1) 所示，元件節點 C 位於骨幹網路，其他元件節點 1、2、3、4、5、6、7、En_1 、En_2 及 En_3 屬於接取網路，其中點 1、3、4、6 及 7 為光纖，節點 2 及 5 為分歧器，En_1 、En_2 及 En_3 為光末端節點之接收器。已知條件：網管人員接收到的告警是 R ={ En_1 ， En_2 }。



圖(1)：模擬之接取網路拓樸。

我們使用樹狀網路多重故障過濾定位演算法四個模組分別執行，可得到如下的結果：

(1) 上游建立模組：

分別對三個光末端節點 En_1 、En_2 及 En_3 建立其上游元件：

$Bin(S_1)$=(1 1 1 0 1 1 0)；

$Bin(S_2)$=(1 1 1 0 1 0 1)；

$Bin(S_3)$=(1 1 0 1 0 0 0)；

(2) 告警收集模組：

假設管理者收到 En_1 及 En_2 發出之告警：

R={ En_1 ， En_2 }⇨ $Alarm$=(1 1 0)；

(3) 元件過濾模組：

經由發出告警之元件，找出可能故障之候選元件：

$CF_1 = Bin(S_1) \vee Bin(S_2)$

=(1 1 1 0 1 1 1)，

而根據沒有發出告警之元件，過濾出正常之元件：

$CF_0 = Bin(S_3)$ = (1 1 0 1 0 0 0)，

所以，

$CF = CF_1 \wedge (\sim CF_0)$

=(1 1 1 0 1 1 1)∧(0 0 1 0 1 1 1)

=(0 0 1 0 1 1 1)

(4) 候選選擇模組：

因元件節點 6 及節點 7 是兄弟節點，而一個元件故障比兩個元件故障之可能性高，所以是祖父節點 3 或是父節點 5 故障之可能性，遠比元件節點 6 及節點 7 同時故障之可能性高；因此，

CS =(0 0 1 0 1 0 0)；

若我們修正成只有父節點 5 故障，而排除祖父節點 3 故障，則可能產生定位錯誤，因為，節點 3 故障，也能導致光末端節點 En_1 及 En_2 同時發出告警，而沒有被定位出來。

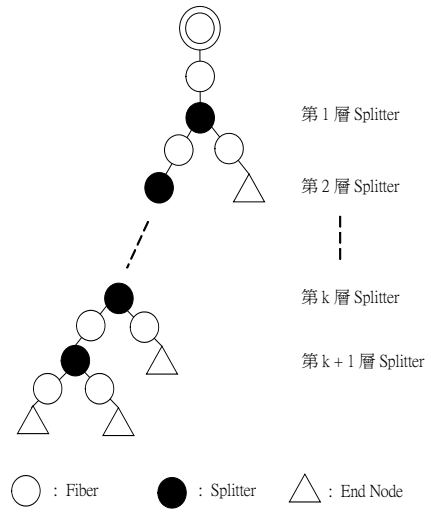
所以，樹狀網路多重故障定位演算法輸出故障元件位置是元件節點 3(光纖)或節點 5(分歧器)。

2.5 演算法複雜度分析

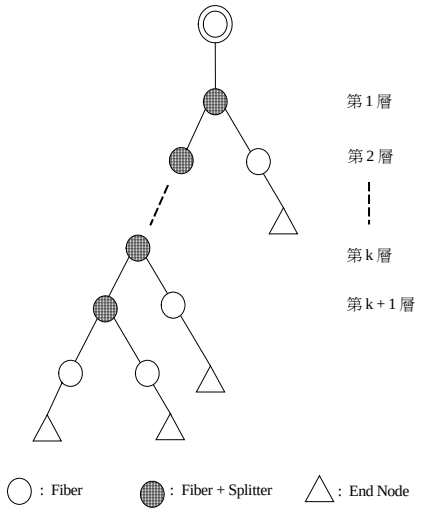
為了比較 FLA 與 TMFLA 複雜度之使用的符號一致性，我們將統一以光末端節點個數 n_a 為變數，所以元件節點個數 n_c ，我們將一律換成 n_a 的函數；因此我們將先證明下列定理。

定理 1：對於網路拓樸是圖(2-a)或圖(3-a)的二進樹的接取網路，其元件節點個數都是光末端節點個數的 3 倍少 2，即 n_c=3*n_a -2。

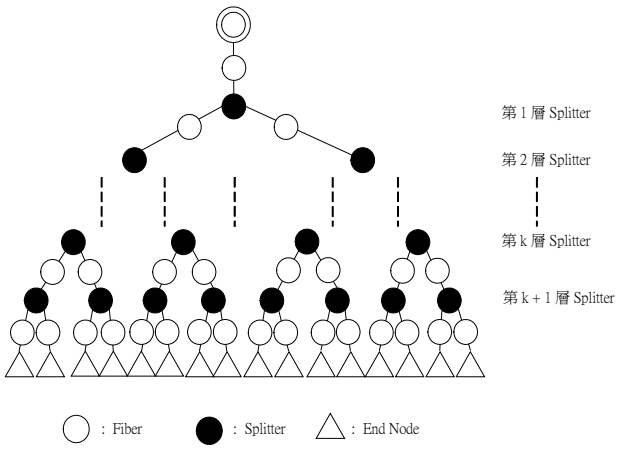
證明：我們將以數學歸納法來分別證明此兩種情形。



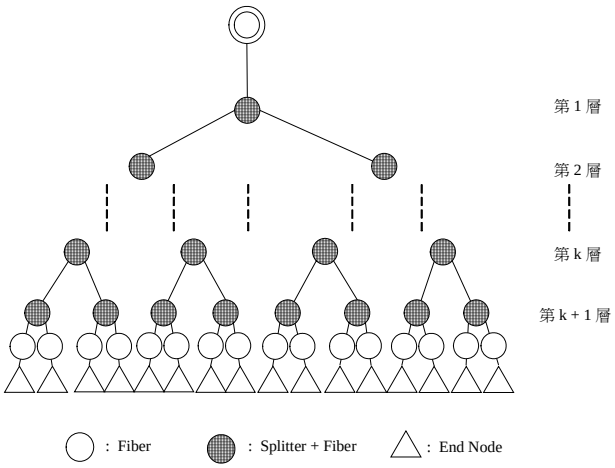
圖(2-a)：二進樹。



圖(2-b)：合併後之純二進樹。



圖(3-a)：二進樹。



圖(3-b)：合併後之滿二進樹。

(1) 對於如圖(2-a)的網路，我們可以改成圖(2-b)之網路拓樸架構(即將每一個分歧器和其上方之光纖合併成一個元件)。

①當有 N=1 層時，網路元件有 n_c=4 個，光末端節點有 n_a=2 個，所以滿足 n_c=3*n_a – 2。

②當有 N=k 層時，若元件有 kc 個，光末端節點有 ke 個，設滿足 kc=3*ke – 2 之關係。

③當有 N=k+1 層時，則元件多了 3 個，即共有 kc + 3 個；而光末端節點則多了 1 個，即共有 ke + 1 個；則元件個數 kc + 3 = (3*ke – 2)+3 = 3*ke+1 = 3*(ke+1) – 2；即有 N=k+1 層時，元件個數為光末端節點的 3 倍少 2。#

(2) 對於如圖(3-a)的網路，我們可以改成圖(3-b)之網路拓樸架構(即將每一個分歧分歧器和其上方之光纖合併成一個元件)。

①當有 N=1 層時，網路元件有 n_c=4 個，光末端節點有 n_a=2 個，所以滿足 n_c=3*n_a – 2。

②當有 N=k 層時，若元件有 kc 個，光末端節點有 ke 個，設滿足 kc =3*ke – 2 之關係。

③當有 N=k+1 層時，則元件多了 $3*2^k$ 個，即共有 kc + $3*2^k$ 個；而光末端節點多了 ke 個，即共有 2*ke 個；則元件個數

kc + $3*2^k$ = (3*ke–2)+ $3*2^k$ = 3*(ke+ 2^k)-2 = 3*(ke + ke)-2 = 3*(2*ke)-2；

即有 $N=k+1$ 層時，元件個數為光末端節點的 3 倍少 2。#

由上述之定理 1，我們有 $n_c = O(n_a)$，因此複雜度 $O(n_c * n_a)$，可以等於是 $O(n_a^2)$。

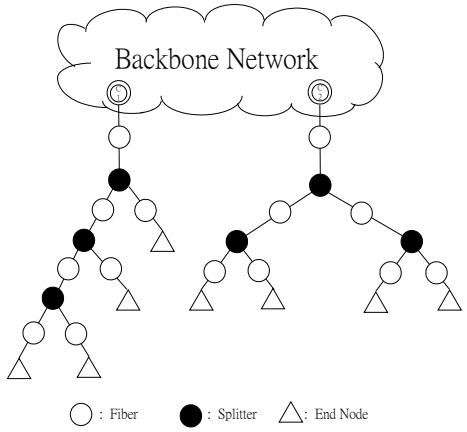
接下來討論演算法之複雜度。故障定位演算法（FLA）機制，包含了事先計算階段(PCP)及搜尋階段。其中 PCP 幾乎佔了大部分之計算量，它的複雜度為 $O(4^{n_a})$[4]；而搜尋階段只是比對告警二元向量與資料庫中二元樹，所以它的複雜度為 $O(n_a)$。因此故障定位演算法(FLA)整體複雜度為 $O(4^{n_a}) + O(n_a) = O(4^{n_a})$，這是一個成長非常可怕的指數複雜度。

而樹狀網路多重故障過濾定位演算法(TMFLA)中有四個工作模組，分別為「通道建立」、「告警捨棄」、「元件過濾」及「候選選擇」。其中「通道建立」模組，須建構接取網路上的所有光末端節點對所有元件的上游源頭，所以複雜度為 $O(n_c * n_a) = O(n_a^2)$；「告警捨棄」模組，當網管收到光末端節點所發出之告警後，將告警轉換成二元向量，故其複雜度為 $O(n_a)$；「元件過濾」模組，是對所有光末端節點的上游源頭分別做 point-wise OR 運算，再做 point-wise AND 運算，所以其複雜度為 $O(n_c * n_a) = O(n_a^2)$；「候選選擇」模組，是根據網路拓樸在同一時間，n 個元件之故障比 n+1 個之故障的可能性高而做調整的故障可能性，所以其複雜度為 $O(n_a)$。因此樹狀網路多重故障過濾定位演算法(TMFLA)的整體複雜度為 $O(n_a^2) + O(n_a) + O(n_a^2) + O(n_a) = O(n_a^2)$。

三、模擬及效能評估

3.1 模擬環境

我們採用 5 種網路拓樸架構為類似圖(4)中之骨幹網路及兩個接取網路，而光末端節點內部為一接收器。



圖(4)：左 8 階、右 6 階之二進樹的網路拓樸架構。

兩個接取網路分別為：

case 1：左 6 階、右 4 階之二進樹架構；

case 2：左 6 階、右 6 階之二進樹架構；

case 3：左 8 階、右 6 階之二進樹架構；

case 4：左 8 階、右 8 階之二進樹架構；

case 5：左 10 階、右 8 階之二進樹架構。

在 5 種 case 中，都各取 25 種發生不同之告警情況來模擬。我們採用 MATLAB 軟體來進行數值模擬。

3.2 FLA 及 TMFLA 之定位結果比較

我們分別以 FLA 與 TMFLA 兩種演算法來做模擬。模擬結果：故障定位演算法(FLA)與樹狀網路多重故障過濾定位演算法(TMFLA)，對 5 種 case 中的各 25 種發生不同之告警情況所做的故障元件之定位，得到的故障之候選元件集合，結果全部都相同。

3.3 程式執行時間之比較

我們分別以 FLA 與 TMFLA 兩種演算法來做模擬，比較建立資料庫之時間及搜尋比對故障元件位置之時間。

3.3.1 建立資料庫時間之比較

當網路拓樸架構分別為 case 1、case 2、case 3、case 4 及 case 5 時，我們首先使用 FLA 演算法來模擬，結果在建立每一個資料庫之花費時間分別為

0.14 秒、0.63 秒、1.66 秒、3537 秒及 18864 秒，如表(一)所示。

表(一)：FLA 與 TMFLA 在 5 種 case 之二進樹的拓樸網路中，建立資料庫所花的時間比較（單位：秒）

Case	1	2	3	4	5
FLA	0.14	0.63	1.66	3537	18864
TMFLA	0	0	0	0	0

而當我們使用 TMFLA 演算法來模擬，不管是 case 1、case 2、case 3、case 4 或 case 5 時，都不需要時間來建資料庫，亦如表(一)所示。

由表(一)之模擬結果顯示：

(1) FLA 之建構資料庫時間，會隨著網路拓樸之規模(即二進樹之階數或光末端節點)增加，而非常快速地增加其演算時間，這剛好也驗證了其複雜度是 $O(4^{n_a})$ 之論點。

(2) TMFLA 不必事先建構資料庫，故其時間都是 0。

3.3.2 搜尋比對故障元件位置之時間

當網路拓樸架構分別為 case 1、case 2、case 3、case 4 及 case 5 時，在每一種 case 中，我們各取 25 種發生不同之告警情況來模擬，並計算此 25 種情況之搜尋比對的平均花費時間，以達到最小之變動量，務求穩定之搜尋定位時間。

我們首先使用 FLA 演算法，來分別對 5 種 case 進行模擬時，結果得到搜尋比對之平均花費的時間分別為 40.1 微秒、54.9 微秒、62.4 微秒、99.6 微秒及 111.9 微秒，如表(二)所示。

然後我們再使用 TMFLA 演算法，來分別對 5 種 case 進行模擬時，結果得到搜尋比對之平均花費的時間分別為 73.4 微秒、109.6 微秒、131.8 微秒、211.8 微秒及 226.6 微秒，如表(二)所示。

表(二)：FLA 與 TMFLA 在 5 種 case 之二進樹的拓樸網路中，搜尋比對所花的平均時間比較（單位：微秒）

Case	1	2	3	4	5
FLA	40.1	54.9	62.4	99.6	111.9
TMFLA	73.4	109.6	131.8	211.8	226.6
TMFLA /FLA	1.8	2.0	2.1	2.1	2.0

由表(二)之模擬結果顯示：

(1) FLA 之搜尋比對所花的平均時間，會隨著網路拓樸之規模(即二進樹之階數或光末端節點)增加，而緩慢地增加其演算時間，這剛好驗證了其複雜度是 $O(n_a)$ 之論點。

(2) TMFLA 搜尋比對所花費的平均時間，會隨著網路拓樸之規模的增加，而增加其演算時間，這剛好驗證了其複雜度是 $O(n_a{}^2)$ 之論點。

(3) TMFLA 的搜尋比對所花費的平均時間約為 FLA 的 1.8 至 2.1 倍。

四、結論

在本論文中，對於網路拓樸是樹狀之接取網路的多重故障元件之定位，我們提出樹狀網路多重故障過濾定位演算法（TMFLA），用來改善既有的故障定位演算法(FLA)的演算時間，尤其，此演算法特別適用於網路拓撲或通道需經常性變更時的情形。

我們的擬結果顯示：

(1) 樹狀網路多重故障過濾定位演算法（TMFLA）與故障定位演算法(FLA)對於樹狀架構之網路拓撲的多重故障定位之候選元件集合，結果完全相同。

(2) 我們所提出的樹狀網路多重故障過濾定

位演算法（TMFLA），完全不需先建立資料庫，因而大大的降低了記憶體空間的需求及演算時間。

（3）樹狀網路多重故障過濾定位演算法（TMFLA）之搜尋比對的平均時間約為故障定位演算法(FLA)的 2 倍左右。這是因樹狀網路多重故障過濾定位演算法完全不需建構任何的資料庫，因此在搜尋比對時間方面所付出的代價。

五、參考文獻

[1] 陳奇書, “在分波多工網路上故障元件定位演算法之設計”, 高雄第一科技大學電腦與通訊工程研究所, 碩士論文,2003.

[2] C. Mas, “Fault Localization at the WDM layers ”, *Kluwer Academic Publishers* , 1999.

[3] C. Mas and P. Thiran , “An Efficient Algorithm for Locating Soft and Hard Failures in WDM Networks” , *IEEE Journal on selected areas in Communications, Management Symposium*, Vol.18, no.10, pp.1900-1911, October 2000.

[4] C. Mas, “Fault Location Algorithms for Optical Networks”, *Ph. D. dissertation 2164*, EPFL, 2000.

[5] D. Zhou and S. Subramaniam, “Survivability in optical networks”, *IEEE Network*, pp. 16-23, November/December 2000.

[6] L. Sahasrabuddhe , S. Ramamurthy , B. Mukherjee. “Fault Management in IP-over-WDM Networks：WDM Protection Versus IP Restoration”, *IEEE Journal on selected areas in communications*, Vol. 20, no.1, pp.21-33, January 2002.

[7] N. S. V. Rao, “Computational Complexity Issues in Operative Diagnosis of Graph-based Systems”, *IEEE Trans. Computers*, Vol.42, pp.447 - 457, Apr.1993.

[8] R. Ramaswarmi , and K. N. Sivarajan , *Optical Networks*, 2th edition, Morgan Kaufmann, San Francisco, 2000