適用於工廠無線感測網路的強健傳輸排程演算法的設計

丁品熏[a]、胡仁豪*[a]、侯廷昭[b]
國立中正大學通訊工程研究所[a]
國立中正大學通訊工程學系及電信研究中心[b]

摘要 —在工廠無線感測網路中，為了讓監控設備的感測節點其感測資料能盡快到達監控中心，以便針對出現問題的設備及時做處置，需保證網路時間延遲應在可接受的時間延遲上限 (Delay Bound) 內。本論文研究鏈狀結構的傳輸排程演算法，藉由令牌 (Token) 在鏈上的傳遞，安排每一節點傳輸的順序，避免碰撞的發生，也提供時間延遲的保證。每條子鏈上的節點順序即為節點傳送資料封包的順序，但是資料封包傳送的路徑則是依據 Sink Tree 上的最短路徑，以降低時間延遲。[1]

一、 簡介

在無線網路中，IEEE 802.11 利用 CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) 的媒介存取方式以 DCF (Distributed Coordination Function) 方式傳輸，在 CSMA/CA 的機制下，網路中節點若要傳輸封包，必須先監聽媒介的狀態，若是媒介處於空閒的狀態下便可以傳輸，若是處在忙碌的狀態，必須先延遲一段時間等待媒介空閒，之後再等待一個隨機退後時間(Random Backoff Time) 便可傳輸封包，此種媒介存取方式能夠降低封包碰撞的機率，但由[1]、[2]得知，當網路節點數增加時，碰撞機率也會逐漸增加，導致整體效能降低，且封包的傳輸延遲時間也隨之增加。在無線網路中，若在高網路負載情況下，TDMA 提供優於以競爭為基礎的協定方式，解決節點因競爭通道造成的封包碰撞及延遲，並且有較高的吞吐量 (Throughput)及具有上限的時間延遲 (Bounded Latencies)，這項優點讓 TDMA 適用於時間敏感 (Time-sensitive) 的應用中，例如自動控制的監測系統等。考量將無線感測網路運用在工廠設備的監控[3]，由於工廠環境具有高時間敏感度的需求，以應對機械設備隨時可能發生的故障，而導致生產力的下降，基於 TDMA 的觀念但又不需要精準同步要求的排程機制，最適合工廠設備監控的環境。因此本論文研究以鏈為基礎之傳輸排程協定，鏈中的節點順序即為節點傳送資料封包的順序，資料封包內夾帶令牌 (Token)，以令牌在節點之間的傳遞達成節點之間的同步。再者構成每條鏈的節點數若固定，則可事先估測封包傳輸時間延遲的上限。

二、 相關研究

在 CTSBD (Chain-based Transmission Scheduling with Bounded Delay)[4]中，此協定在網路中建立鏈狀結構的拓撲，如圖一所示，假設網路中有多個 Sink，Sink 之間的上下游關係已事先決定，在每個上游 Sink 與其下游 Sink 之間將建構出一條子鏈，此子鏈的方向為上游 Sink 指向其下游 Sink 的方向，子鏈中的每個節點由上游 Sink 往下游 Sink 的排列順序即為令牌傳送的順序。當某節點收到來自上游節點的令牌時，該節點才有權力將所有佇列的資料封包傳送完畢，之後才會釋放令牌給下游節點。而每個節點在傳送資料封包之前，會先查看路由表資訊，將資料封包傳送給距離自己最近的 Sink，故資料封包傳送的方向可能會與令牌的傳送方向相反，如圖二所示。

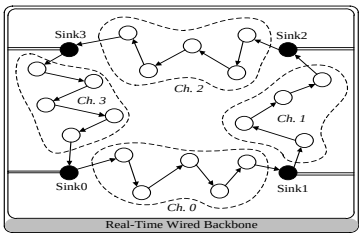


圖 一：CTSBD 網路運作方式

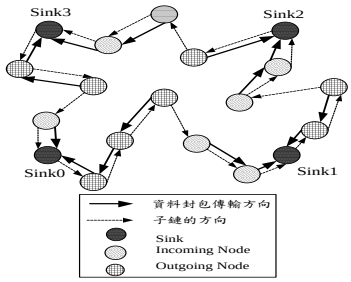


圖 二：子鏈的方向和節點的資料封包傳送方向

本論文研究一個以鏈為基礎的強健傳輸排程協定 (Robust Chain-based Transmission Scheduling with Bounded Delay, RCTSBD)，將對於 CTSBD[4]進行修正。由於先前 CTSBD[4]鏈狀結構的拓撲尚存缺陷，導致上游 Sink 與下游 Sink 之間無法順利建立最初路徑，使得感測資料的傳輸無法順利進行，在此我們對建立最初路徑的尋找、子鏈的建立這兩部份演算法進行修正，讓每個 Sink 與其下游 Sink 之間建構出一條子鏈，此子鏈的方向為上游 Sink 指向其下游 Sink 的方向，在子鏈形成後，會有些節點無法被加入，我們稱這些節點為孤立節點，孤立節點無法加入子鏈，導致無法傳送資料封包到 Sink。我們將探討孤立節點產生的條件及消除孤立節點的作法。

三、 研究議題－資料封包傳輸的排程

3.1 網路架構

在 RCTSBD 使用與 CTSBD[4]相同網路架構，如圖一

[1] 本研究由國科會贊助，計畫編號 NSC 100-2221-E-194-039-MY2。(此處說明研究經費補助單位).

Corresponding author:hu411142@gmail.com
DOI:10.6159/IJSE.2014.(4-1).32

所示，感測節點依需求，隨機佈建在各處，並以無線方式互相 (包含 Sink) 溝通，Sink 節點建置在工廠多處固定位置，收集感測節點的資料，並由後端有線網路將資料傳至中央監控中心。在開始傳送資料封包前的程序，例如 : 建立鄰居清單、路由表的建立、節點的分組以及子鏈的建立，所有節點 (包含 Sink) 均使用相同的通道進行通訊。上游 Sink 與下游 Sink 之間透過有線網路相互溝通。當下游 Sink 接收到令牌時，隨即透過有線網路通知上游 Sink 發送下一個令牌。因此每個 Sink 都需配置兩張 WLAN 網路卡，各自操作在不同的通道上，而其他節點只需配置一張 WLAN 網卡。

3.2 路由表的建立

路由表藉由每個 Sink 廣播 HOP_TREE 封包來建立，此 HOP_TREE 封包包含來源節點 Sink 的位址及離來源節點 Sink 的 Next Hop 和幾個跳數的資訊，Sink 本身的跳數設定為零，當節點接收到 HOP_TREE 封包時，將封包內的跳數加一，便是該節點離特定來源節點 Sink 的總跳數，於是把計算過後的跳數、封包的傳送節點及 Sink 位址記錄在自己的路由表中，再將 HOP_TREE 封包廣播給鄰近的節點。

當網路中每個節點都建立每個 Sink 的資訊 (Sink 位址、跳數及 next_hop) 後，便將本身的路由表和鄰居節點清單告知距離自己最近的 Sink，因此每個節點至距離自己最近的 Sink 間的路徑就確立了，而這路徑將用於傳輸資料封包的用途

3.3 節點的分組

建立 Sink 間的子鏈之前，各個 Sink 將屬於自己 Sink Tree 的子孫 (Offspring) 節點分成二組，分別為進入組 (Incoming) 和外出組 (Outgoing)，進入組是指該子孫節點與上游 Sink 間的距離比與下游 Sink 間的距離較小，反之則為外出組。為了降低系統成本與提升建置彈性，我們假設網路上的節點皆不需知道自己的地理位置，因此分組演算法利用跳數取代地理距離的比較。

使用跳數當作距離可能會出現有些節點到上游 Sink 的跳數與到下游 Sink 的跳數相同的情況，以致於無法確切的為這些節點分組，因此本文提出一種判斷方式來解決這種情況，判斷節點的鄰居，這些鄰居節點中被分為進入組和外出組的個數比例，若是進入組的個數較外出組多，則該節點被分為進入組；反之，則該節點被分為外出組。

3.4 鏈的建立

3.4.1 最初路徑的尋找

分組完畢後，各個 Sink 要將屬於進入組的節點清單告知上游的 Sink，之後各個 Sink 便擁有屬於自己的外出組節點資料以及下游 Sink 告知的進入組節點清單，這些節點將是組成子鏈的成員。此程序先由最低位址的 Sink 發起，此一建立子鏈的發動者 (Initiator Sink) 會傳送 ANNOUNCE 封包給上游的 Sink (Terminator Sink)。

首先 Initiator Sink 先判別自己是否擁有進入組的節點，若 Initiator Sink 擁有進入組的節點，就選擇離自己一個跳數且為進入組的節點當作最初路徑的第一個節點來轉送封包，若 Initiator Sink 無法找到離自己一個跳數且為進入組的節點，再判別是否有擁有離自己兩個跳數的進入組的節點，若 Initiator Sink 擁有離自己兩個跳數的進入組的節點，就去尋找此離自己兩個跳數的進入組節點的鄰居，此鄰居須為外出組的節點並且離 Initiator Sink 距離一個跳數，若找到符合此條件外出組的節點，則將它更改為進入組的節點並當作最初路徑的第一個節點來轉送封包；若是 Initiator Sink 找不到以上符合條件的節點並只擁有外出組的節點時，則從外出組節點中挑選一個離 Terminator Sink 最近的節點當作第一個轉送點，並將該節點的組別變更為進入組，將它更改為進入組的節點並當作最初路徑的第一個節點來轉送封包。

當第一個轉送節點接收到 ANNOUNCE 封包後，將自己的位址記錄在封包中，並且記錄此封包的上一個轉送者位址，接著查看自己的路由表中前往 Terminator Sink 的下一步轉送者是否為封包中進入組的成員，若為進入組成員則傳送封包；反之，檢查是否為封包中外出組的成員，若是外出組成員則搜尋封包內進入組的清單，從中選擇一個是自己的鄰居節點也是離 Terminator Sink 更近的進入組節點作為封包的下一步轉送者；反之，此下一步轉送節點則是屬於 Terminator Sink 的外出組或進入組節點。若自己的路由表中前往 Terminator Sink 的下一步，都無法符合以上條件，則將此前往 Terminator Sink 的下一步轉送者外出組節點更改為進入組節點，這不斷反覆檢查的動作是為確保封包經由最初路徑的前段路徑中繼節點是 Initiator Sink 的進入組成員。封包的轉送者都必須把自己的位址記錄在封包中，並記錄封包的上一個轉送者位址後，才能把封包轉送給下一個節點。

當 Terminator Sink 收到 ANNOUNCE 封包時，記錄封包內的進入組節點清單，接著檢查封包中記錄的每個中繼節點是否存在自己的進入組成員，若存在則將該節點的組別變更為外出組，並且再發起另一個 ANNOUNCE 封包給下一個 Terminator Sink，此程序不斷地反覆執行，直到最低位址 Sink 收到 ANNOUNCE 封包為止，至此，Sink 之間的最初路徑便形成，運作方式如圖三所示。

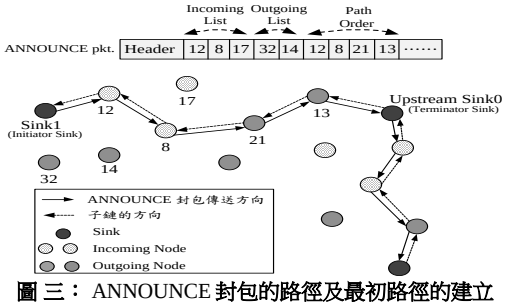


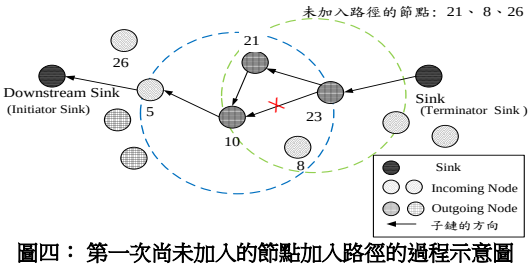
圖 三： ANNOUNCE **封包的路徑及最初路徑的建立**

3.4.2 子鏈的建立

當 Initiator Sink 與 Terminator Sink 之間的最初路徑建立完成後，每個 Sink 將已加入最初路徑的節點與下游 Sink 的進入組節點清單和自己的外出組節點清單作比對，歸納出第一次尚未加入最初路徑的節點清單，並將此清單告知位在最初路徑上的每個節點，接著路徑上的每個

節點依序開始檢查自己的鄰居中是否有未加入的節點，如果有的話，將該節點加入。此程序利用圖四來說明：

當節點 23 從 Sink 接收到第一次尚未加入的節點清單(節點 21、8、26) 後，檢查自己的鄰居，發現鄰居節點 21 和節點 8 在此清單中，首先節點 23 先嘗試將節點 21 加入子鏈路徑中，因此節點 23 詢問子鏈路徑目前的下游節點 10，節點 21 是否為節點 10 的鄰居，結果節點 10 回覆節點 23 為確定節點 21 是節點 10 的鄰居，所以節點 23 將自己傳送令牌的下一步接收者改為節點 21，節點 23 並告知節點 21 其下一步令牌接收者為節點 10，之後節點 23 再嘗試將節點 8 加入路徑，直到節點 23 將能加入路徑的節點加入後，再傳送已加入及未加入的節點清單給下游節點，並輪到下游節點尋找第一次尚未加入的節點。



圖四： 第一次尚未加入的節點加入路徑的過程示意圖

當網路中將第一次尚未加入的節點加入路徑後，每個 Initiator Sink 將屬於自己子孫節點與前往上游 Sink 與前往下游 Sink 路徑比對後，若發現自己節點子孫(包含進入組和外出組)還有未加入的節點，則將這些尚未被加入的節點放置第二次尚未加入的節點清單(進入組節點和外出組節點)，並將第二次尚未加入的外出組節點清單，通知上游 Sink (Terminator Sink)，進行第二次節點加入，此程序利用圖五來說明：

在 Terminator Sink NO. 1 (Sink 1) 得到 Initiator Sink NO. 1 (Sink 2) 的外出組節點 19 和自己的進入組節點 32，歸納出第二次尚未加入的節點清單後，連同 Current PATH 傳給下游節點進行前往下游 Sink 2 的第二次節點加入。

在子鏈建立完成後，還有節點未被加入我們稱這些節點為孤立節點，如圖六(a)是會發生孤立節點 D 的舉例，孤立節點 D 因為無法與現有路徑中的節點 C 互為鄰居，導致無法加入子鏈路徑。所以當有孤立節點發生時，將此孤立節點 D 和孤立節點 D 附近其他節點的通訊範圍增加，使得孤立節點 D 可與節點 C 互為鄰居後，即可被加入子鏈路徑中，如圖六(b)所示。

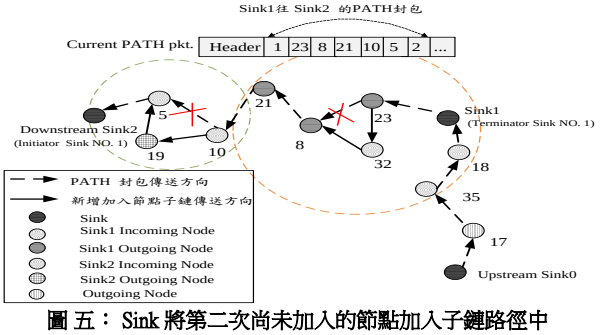
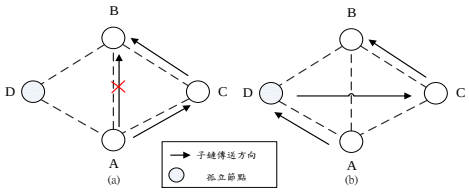


圖 五： Sink 將第二次尚未加入的節點加入子鏈路徑中



圖六：網路中孤立節點發生狀況及解決辦法

3.5 時間延遲上限的估測

時間延遲上限的估測計算方式，以下式子(1)為每條鏈的總時間計算方式，假設每個感測節點在總時間 T 內產生一個資料封包：

$$T = \sum_{j=1}^{|N_m|} h_j \times T_{one\_pkt} = \sum_{j=1}^{|N_m|} T_j \quad (1)$$

$$T_{one\_pkt} = T_{backoff} + DIFS + T_{RTS} + T_{CTS} + T_{DATA} + T_{ACK} + 3 \times SIFS \quad (2)$$

其中，N_m 為子鏈 m 的節點集合，h_j 為 N_m 中節點 j 離 Sink m 的跳數，T_j 為節點 j 傳完自己 buffer 中所有封包所需的時間。

考慮最糟的情況，假設 Sink Tree 的樹葉 (Leave) 節點 i 在第一回合傳完資料封包，令牌交給下一個節點後，又產生自己的資料封包，則此封包必須先在節點 i 的 buffer 中等待$T_{S,i} = T - T_i$的時間單位，到了第二回合，節點 i 才將此封包傳送給下一個轉送點 $i-1$，並且考慮此節點 i 傳送資料封包到 Sink 的路徑剛好是循著子鏈的節點回到 Sink，此情況將有最大的時間延遲。假設封包在節點 i 的排程延遲、佇列延遲、傳輸延遲分別為 $T_{S,i}$、$T_{Q,i}$ 和 $T_{TX,i}$。

$$T_{S,i} = T - T_i \quad (3)$$

$$T_{Q,i} = x_i \times T_{one\_pkt} \quad (4)$$

$$T_{TX,i} = T_{one\_pkt} \quad (5)$$

其中 x_i 為節點 i 需要幫忙轉傳的封包數，也等於節點 i 的下游節點個數。所以封包在節點 i 產生後，到抵達節點 $i-1$ 的延遲為 $T_{L,i} = T_{S,i} + T_{Q,i} + T_{TX,i} = T - T_i + (x_i + 1) \times T_{one\_pkt} = T$，同理，對於中間節點 k，$k = 1,2,\cdots,i-2$，

$$T_{L,k} = T_{S,k} + T_{Q,k} + T_{TX,k} \quad (6)$$

$$= T - T_{k+1} \quad (7)$$

所以封包自節點 i 產生到抵達 Sink 的總延遲為

$$T_{maxDelay} = \sum_{k=1}^{h_i} T_{L,k} \quad (8)$$

$$= h_i \times T - T_i - \cdots - T_3 - T_2 \quad (9)$$

因此，當我們確定 Sink Tree 的拓樸，我們可以知道最大的 h_i，以及 x_k、T_k $(k = i, i-1, \cdots, 1)$，進而估算出 $T_{maxDelay}$。

3.6 吞吐量的估測

吞吐量的估測計算方式，下式(10)為每條子鏈的總週期時間計算方式，並假設每個節點每 100ms 產生一個資料封包需要傳送，每個節點使用 1Mbps 的速率來傳送資料封包：

$$T = N_n \times T_k + \sum_{i=1}^{|N_n|} h_i \times [\ T_{other} + \ T_{Data}] \quad (10)$$

$$T_{other} = \ T_{backoff} + DIFS + T_{\ RTS} + T_{CTS} + T_{ACK} + 3 \times SIFS \quad (11)$$

其中，T_k為 Token 的傳輸時間，T_{Data} 為每個資料封包傳輸時間，N_n為子鏈 n 的節點集合，h_i為 N_n中節點 i 離自己最近的 Sink 的跳數 (Sink Tree Hop)。

$$Throughput = \frac{N_n \times T_{Data} \times 1\ Mbps}{Max\ (T\ ,\ 100ms)} \quad (12)$$

四、 模擬環境與結果

本論文的環境架構是考慮在 30m×30m 的環境中擺放 4 個 Sink，這 4 個 Sink 的位置固定，而網路中其它節點的位置是使用隨機分佈擺放，假設無線媒介為理想通道，不考慮封包傳送 Bit Error，只有碰撞情況下才有封包傳送失敗，並假設每個節點每 100ms 產生一個資料封包需要傳送，每個節點使用 1Mbps 的速率來傳送資料封包，其大小為 250bytes，模擬時間為 60 秒。

下表 I 為模擬參數設定:

表 I 模擬參數設定

Parameter	Value	Parameter	Value
Communication radius	8 m、9m、10m、11m	Slot time	50 us
PHY Header	128 bits	CWmin	16
RTS	288 bits+ PHY Header	SIFS	28 us
CTS	240 bits+ PHY Header	DIFS	128 us
ACK	240 bits+ PHY Header		

4.1 子鏈上節點數對時間延遲的影響

觀察每次在子鏈建立後，依子鏈上的節點數不同時間延遲狀況如圖七所示，在子鏈上若有較多的節點數時，則發生較高時間延遲的節點數也就增加，進而影響整條子鏈的平均時間延遲，相對的在子鏈上若有較少的節點數時，則發生較高時間延遲的節點數也就間接減少。

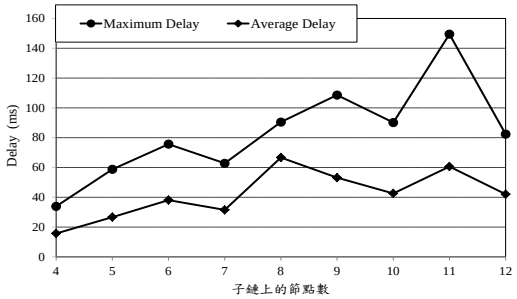


圖 七：子鏈上不同節點數的時間延遲

4.2 子鏈節點數對吞吐量的影響

在網路中隨機擺放節點數為 35 個 (不包含 4 個 Sink)，形成的子鏈，為 sub-chain 0 有 10 個節點數，sub-chain 1 有 6 個節點數，sub-chain 2 有 7 個節點數，sub-chain 3 有 12 個節點數，並假設每個節點在時間 1 秒產生 10 個封包數需要傳送，每 100ms 觀察每條子鏈的吞吐量，觀察 50 次，在此模擬中，如圖八所示，sub-chain 2 所產生的吞吐量為 140kbps，與子鏈上節點數有關，sub-chain 3 所產生的最大吞吐量為 260kbps，與子鏈上節點數和節點的平均跳數有關。

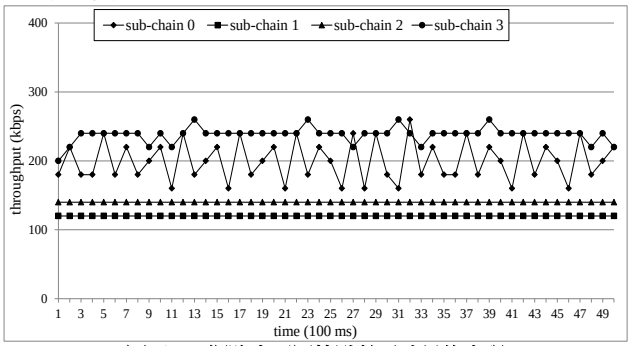


圖 八 ：網路中不同節點數吞吐量的表現

4.3 網路中節點數對傳送比率的影響

如圖九所示，CTSBD [4] 所形成的網路中會有未被加入子鏈的節點 (孤立節點)，因為孤立節點無法傳輸資料封包到達 Sink，導致封包遺失。而 RCTSBD 將這些孤立節點加入子鏈路徑中，讓網路上每個節點都有機會傳送資料封包到達 Sink。

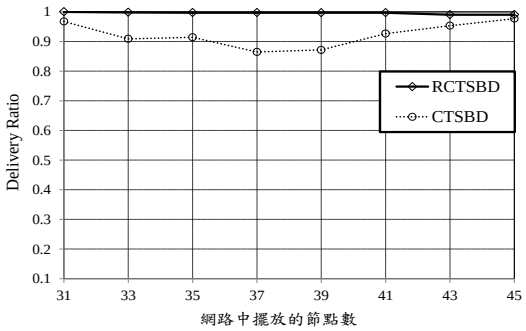


圖 九 ：網路中不同節點數傳送比率的表現

五、 結論與後續研究

本篇論文所提出 RCTSBD 對於先前 CTSBD[4]的傳輸排程進行改良，網路中節點的位置為隨機分佈，易造成節點分佈不均勻，導致建立最初子鏈路徑後有孤立節點的存在，經由 RCTSBD 將孤立節點加入子鏈路徑解決孤立節點問題，也因為將這些孤立節點加入子鏈路徑中，使子鏈節點數增加，吞吐量和 Delivery Ratio 也獲得提升。

六、 參考文獻

[1] G. Bianchi, “IEEE 802.11-Saturation Throughput Analysis,” IEEE Commun. Lett., vol2, pp. 318-320, Dec. 1998.

[2] G. Bianchi, “Performance analysis of the IEEE 802.11 Distributed Coordination Function,” IEEE J. Sel. Areas Commun., pp. 535-547, Mar. 2000.

[3] E. Toscano, L. Lo Bello, “A novel approach for data forwarding in Industrial Wireless Sensor Networks,” Emerging Technologies and Factory Automation (ETFA) , 2010.

[4] 張心芳、丁品熏、侯廷昭, “適用於工廠無線感測網路中具延遲界限之傳輸排程協定,” NST 全國電信研討會 , 2012.