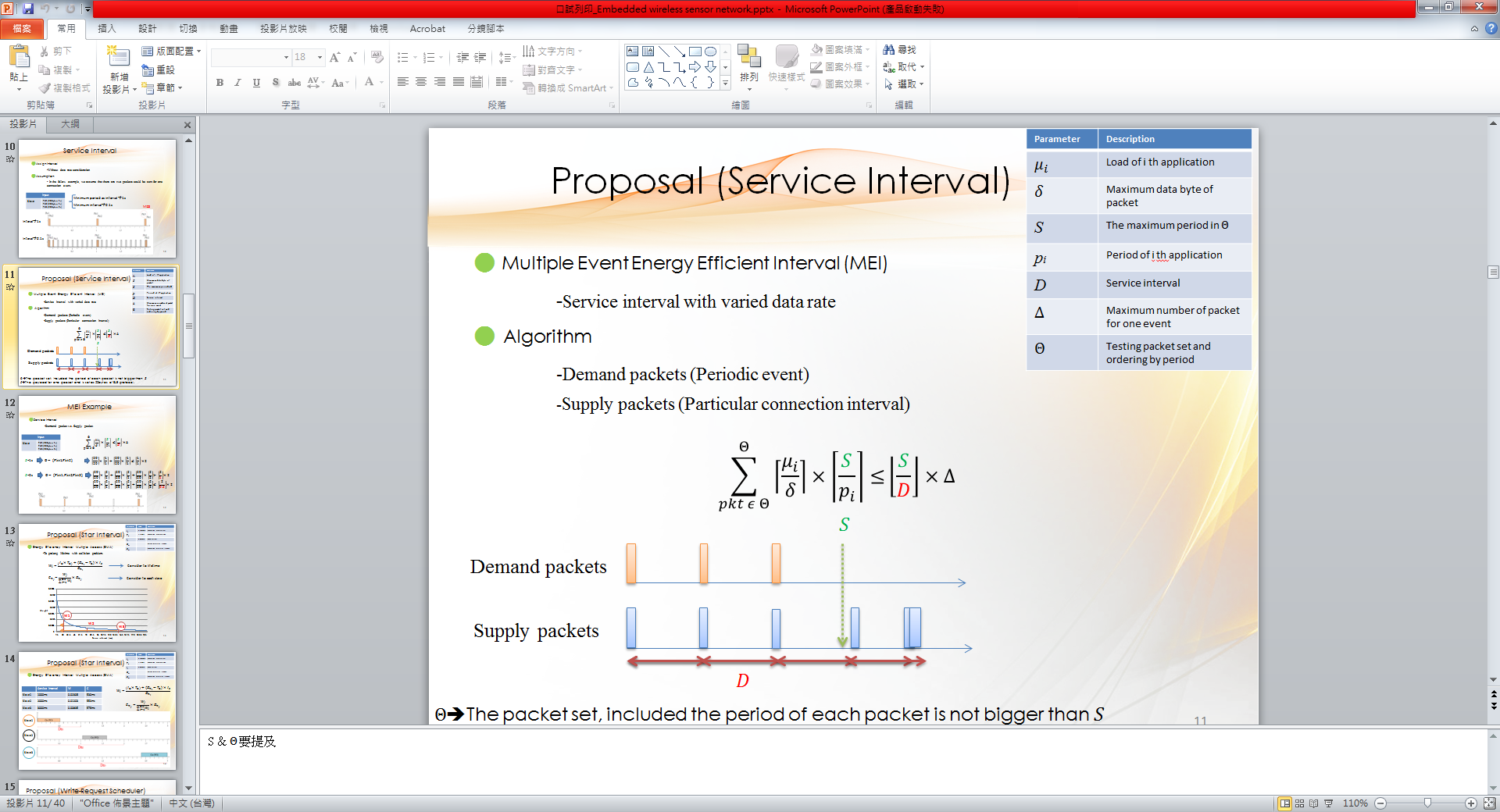
**論文內容**

●系統架構:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
|  | Load of i th application |
|  | Maximum data byte of packet |
| *S* | The maximum period in |
| *pi* | Period of i th application |
| *D* | Service interval |
|  | Maximum number of packet for one event |
|  | Testing packet set and ordering by period |

●**MEI** : 處理不同data rate下對單一node上interval計算，此interval為服務所有application最緊的時間區間 (Service Interval)



●**EIMA** :

概念:利用average current曲線作為各個slave的weight assignment曲線，主要是用此weight表達會造成的耗能多寡。用佔據總體weight的比例去做connection interval計算

若比例高表示一開始一開始耗能高，所以decrease量不能多，否則耗能更高

若比例低表示一開始一開始耗能少，decrease量要多以免造成太多blocking time(占用時間變少)

●**EIF** :

將MEI的service interval當成period，將EIMA的connection interval作為exec time，與EDF方法相似，但會將block考慮進去，若服務此slave會造成他人被block，那麼就寧願等待。

Floor(deadline/conn)\*conn = 會來幾次event\*時間 = 下一次event來的時間

判斷若小於t + cnj 代表 i 會被j 擋住，導致data無法傳輸要累積到下一次event。

●實驗設定:

在不同的data rate下做不同的load比較

(各個slave的period分別為200ms, 400ms, and 800m左右)

●數據分析: Meet 與 life 是相對的，此高彼短

* **Meet ratio:** 以load角度看，MEI🡪demand pkt，Greedy不考慮load(都是調到最小)。EIMA在高rate會接近Greedy。LDC low prio會被delay到後面，在高rate下，RR較LDC高。
* **Life time:** 看connection interval，連傳的次數。 LDC(各自節點週期除節點數)連傳績較Greedy(最小週期除節點數)少(LDC容易被block)，較省電。Polling有優先權概念，高prio次數相對較多，生命週期較RR差。

**模擬內容**

●檔案區分

Energy在write\_request時計算。

Miss在write\_request以及BLE\_EDF(傳輸)的check\_pkt中會計算，最後在寫輸出檔時會整理計算。

●Read Input File

. readsetting=1 -> 由本地端讀取input

. 1.複製.\WSN\_simulation\WSNresult\Default\node3 > .\WSNresult

. 2.設定readsetting=1，設中斷點至main不讓他執行，只要產生exe檔就好

. 3.至剛剛複製的node資料夾中，將每個方法的setting.txt做讀取路徑及方法設定，\WSN\_simulation\GENresult\input\_varied\_node3 (Gen的input位置)

. 4.複製exe至方法資料夾中，直接執行即可

. 5.全部跑完後，Mainparse.py會將FinalResult的資料整理成txt(GNU畫圖用)及cvs(整理每個方法的效能)

. 6.若要跟實作match，模擬practice設定true，則會在計算power那邊I\_sleep=400，會接近實作數據，但是考慮最悲觀情況，因實際上不是每個時間都是400uA，而是1~400間跳動。

●Service Interval (Single node)

**🡪Multiple Event Energy Efficient Interval (MEI)**

校正Service Interval

Supply packet

在S時間點之前

系統提供的pkt數量

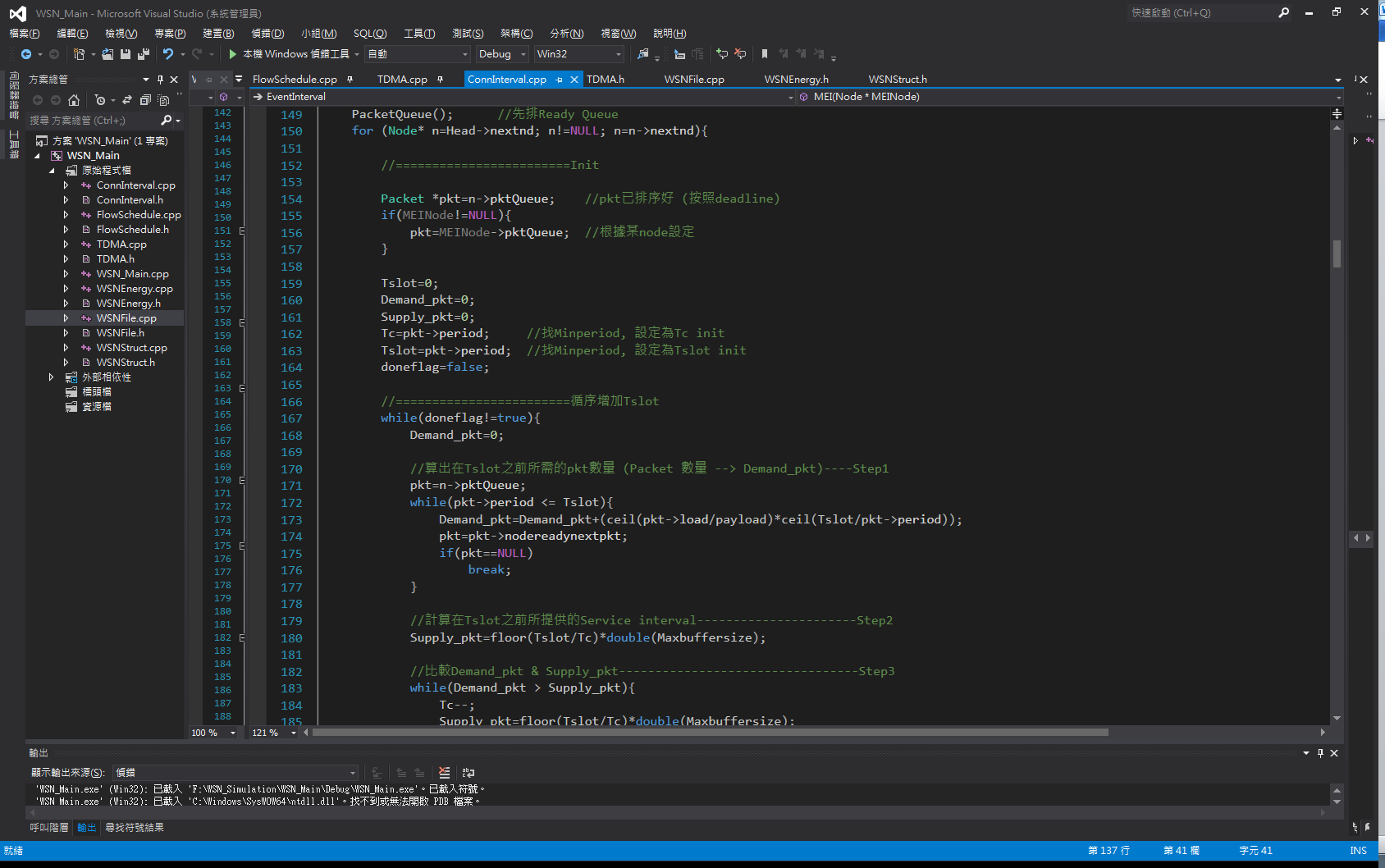
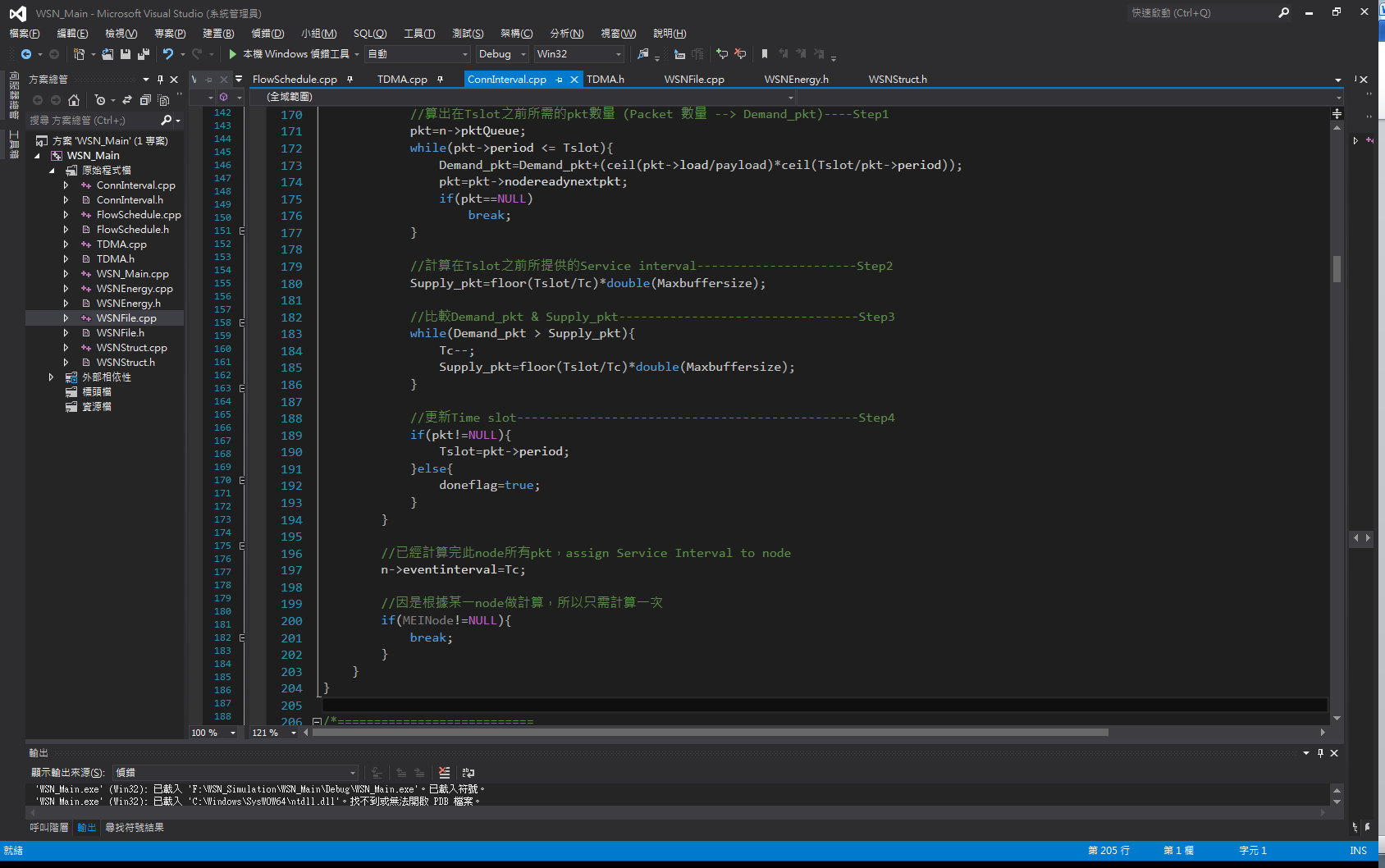
Demand packet

在S時間點之前會有的pkt數量

要先排好優先權

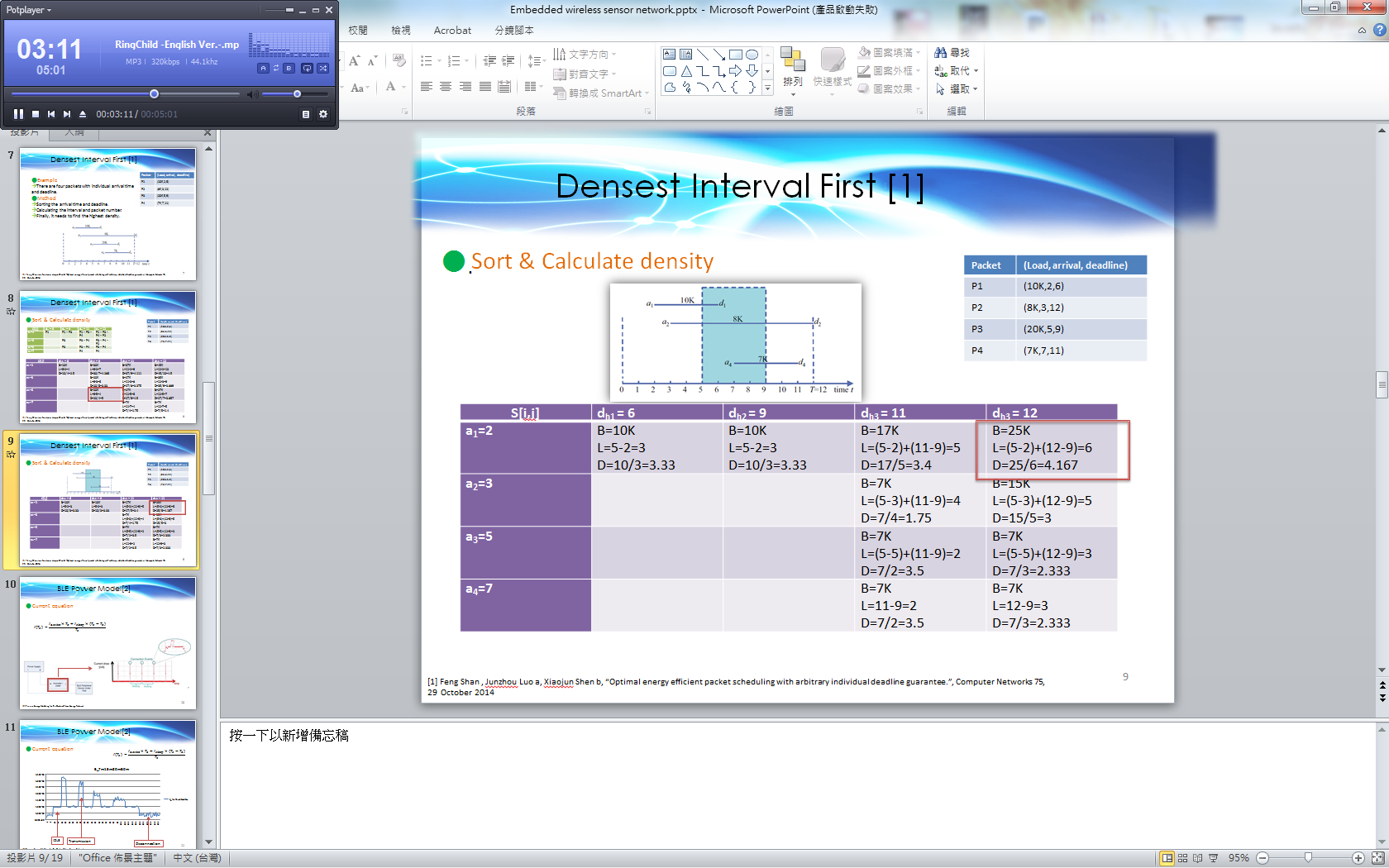
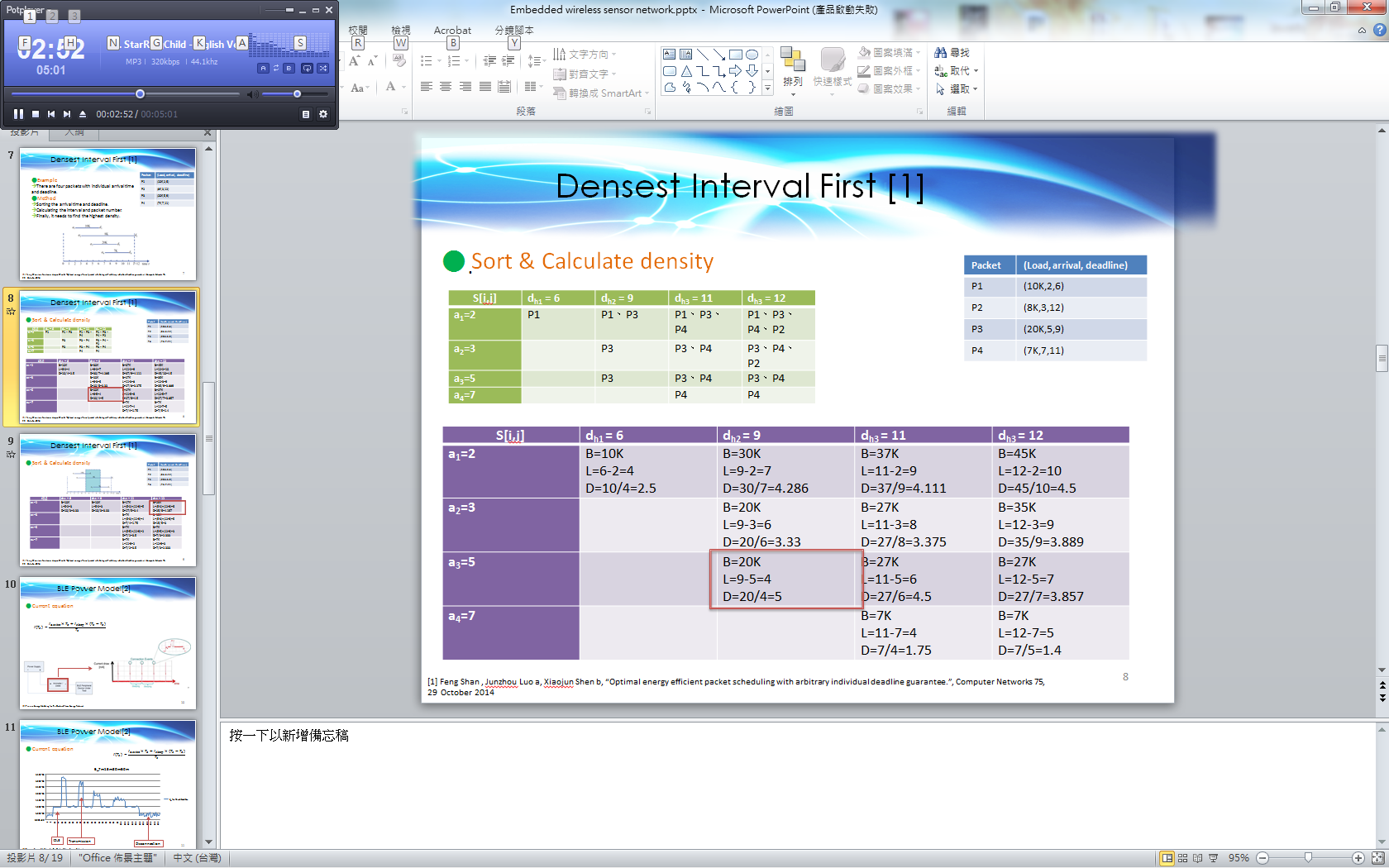
設定好起始Service Interval

S (Tslot)基準點



**🡪Densest Interval First (DIF)**

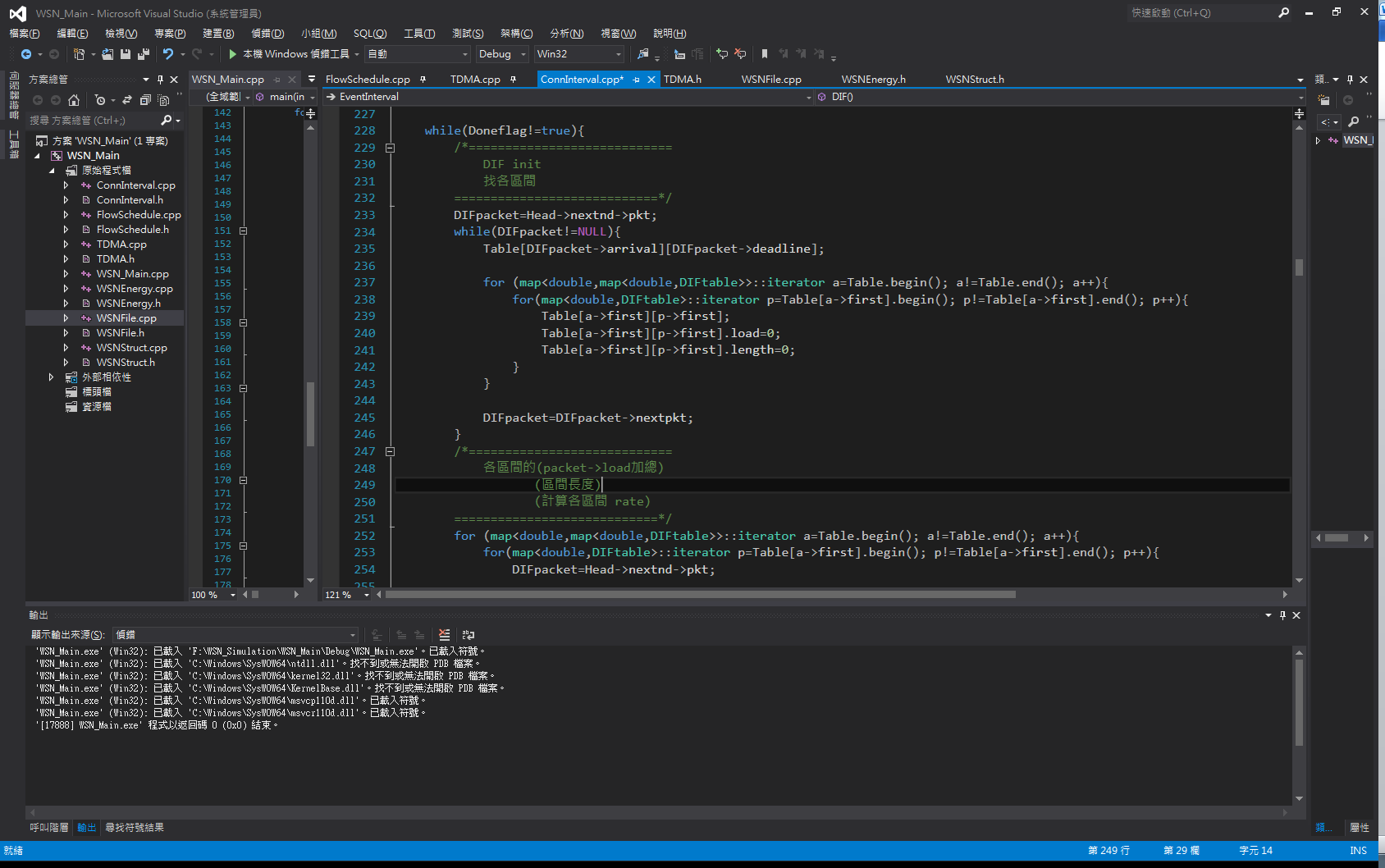
概念: 區間內找最大rate做assign，在assign 長度時要排除已經assign過的區間，區間內的封包定義為封包arrival大於區間起始點，deadline小於區間終點



Map🡺為key對應value或struct的架構 (map<key, value>)

Map->first為key

Map->second為value或struct

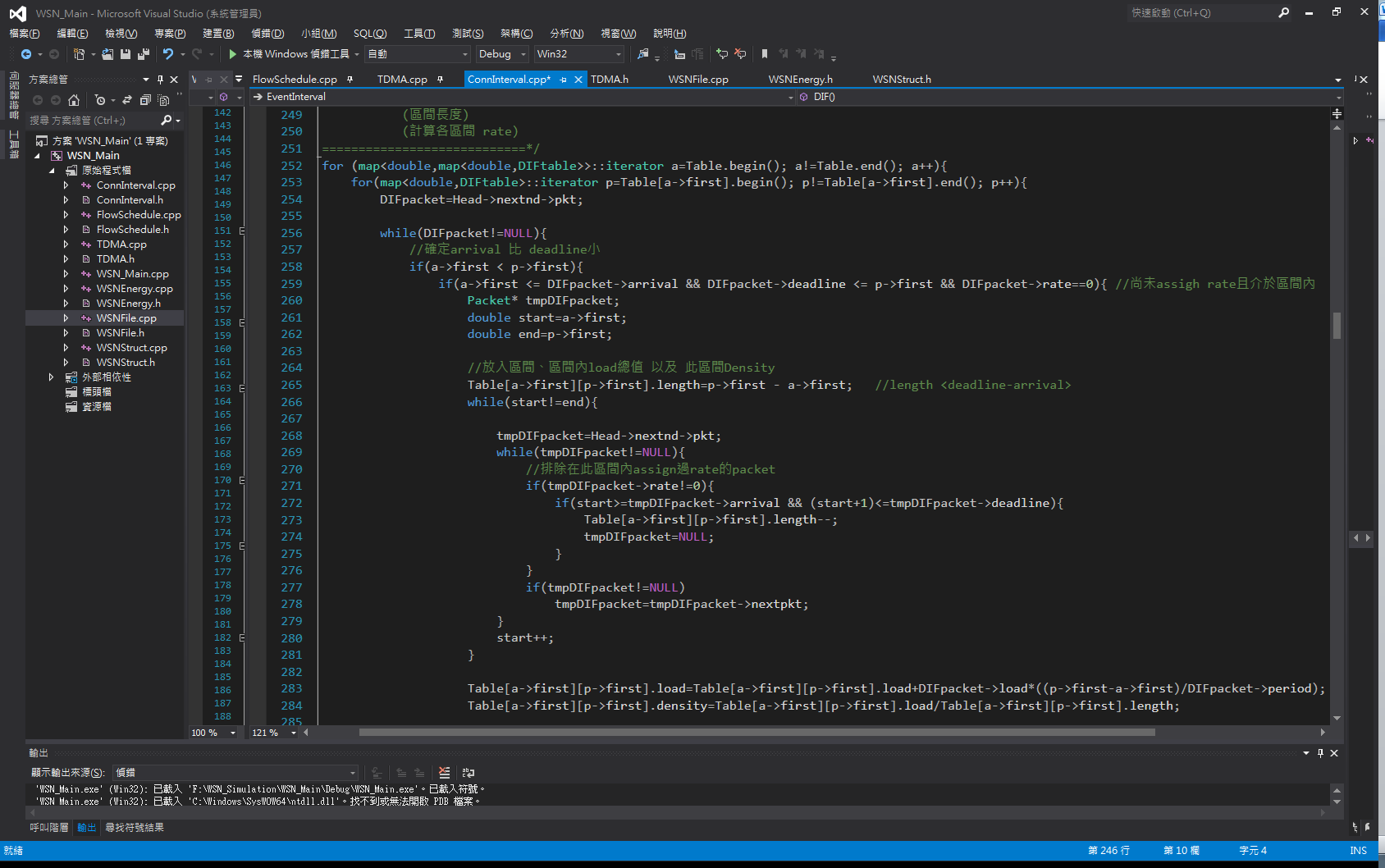


Init

二維map

(**first表示key**, second表示struct)

(Table[arrival][deadline])

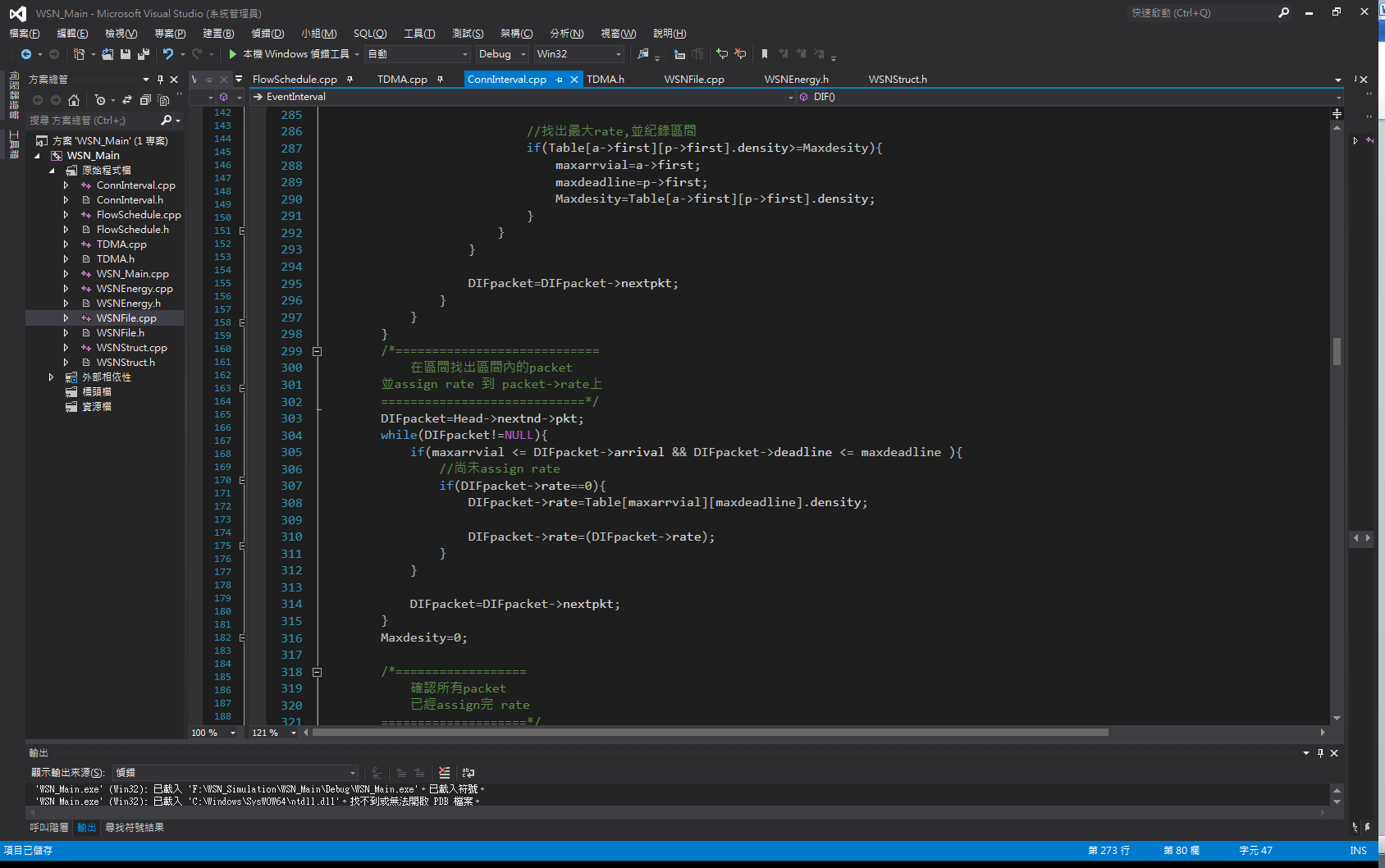


刪除已assign過rate的區間

a->first為arrival

p->second為deadline

(確認DIFpacket在區間內)



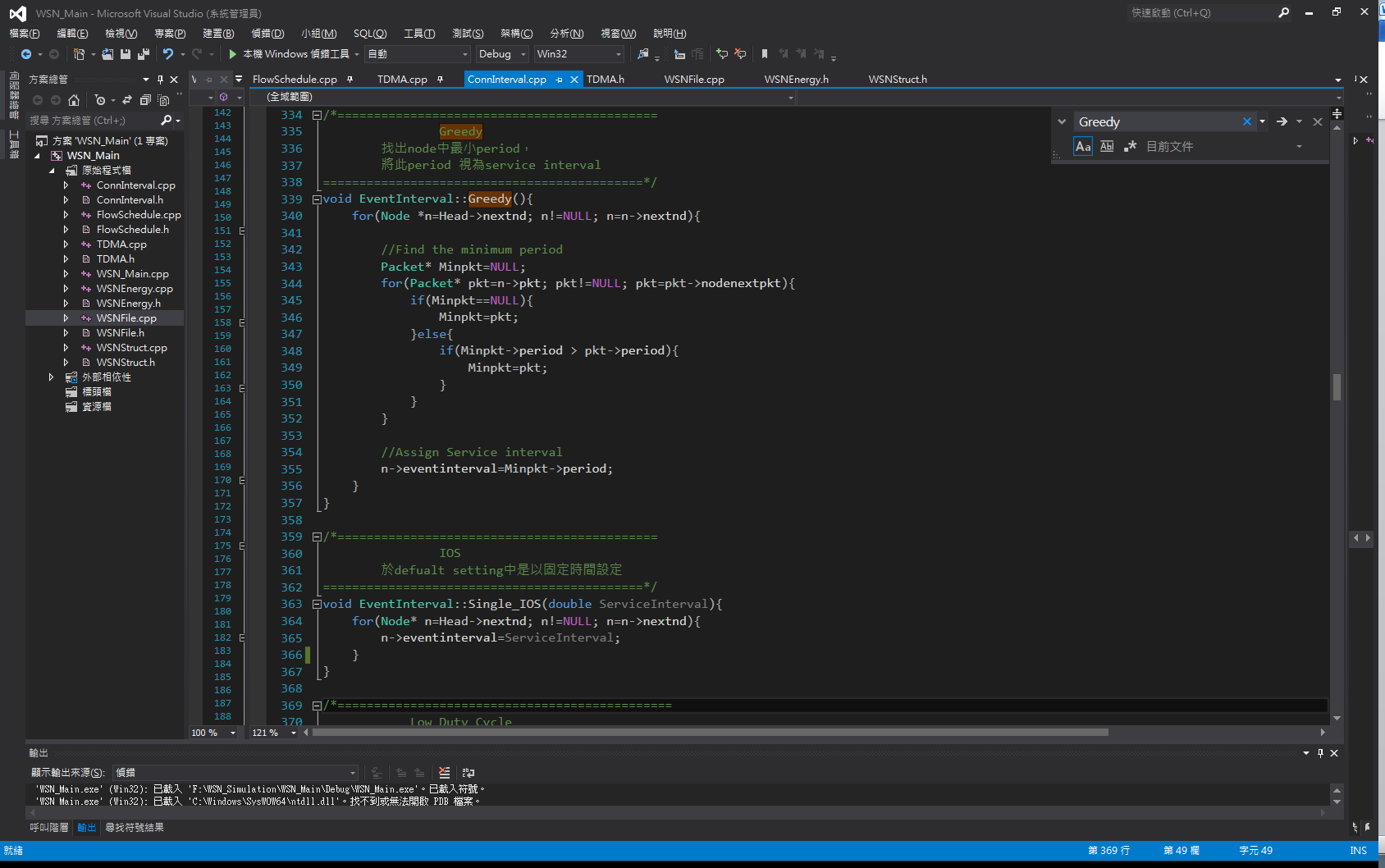
最大rate作各個

packet rate assignment

**🡪Greedy & IOS**

概念: Greedy為採用最小period作為service interval

概念: IOS為採用40ms



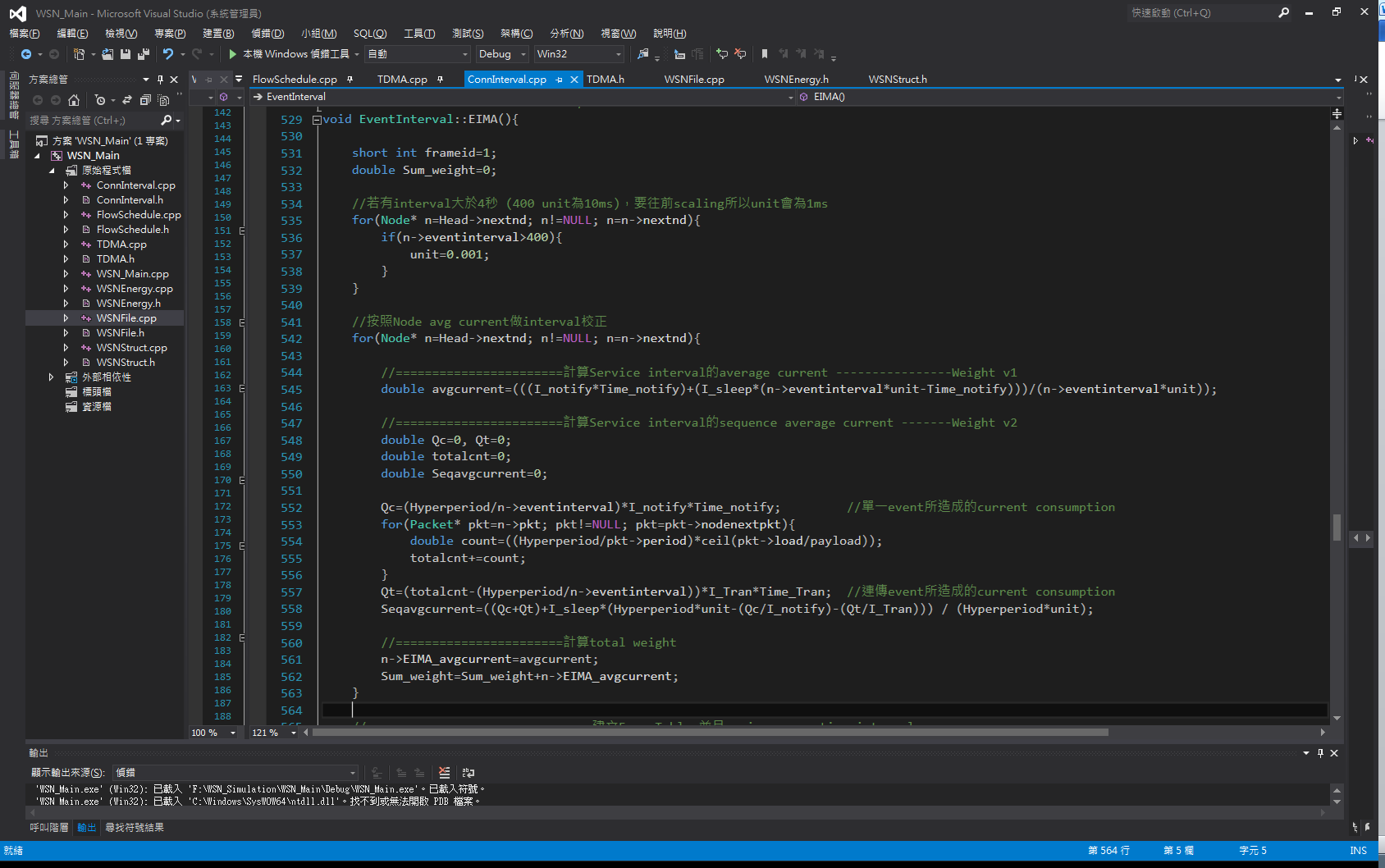
找最小period

Service Interval為30ms

●TDMA

●Connection Interval (Multiple node)

**🡺Energy Efficiency Interval Multiple Access (EIMA)**



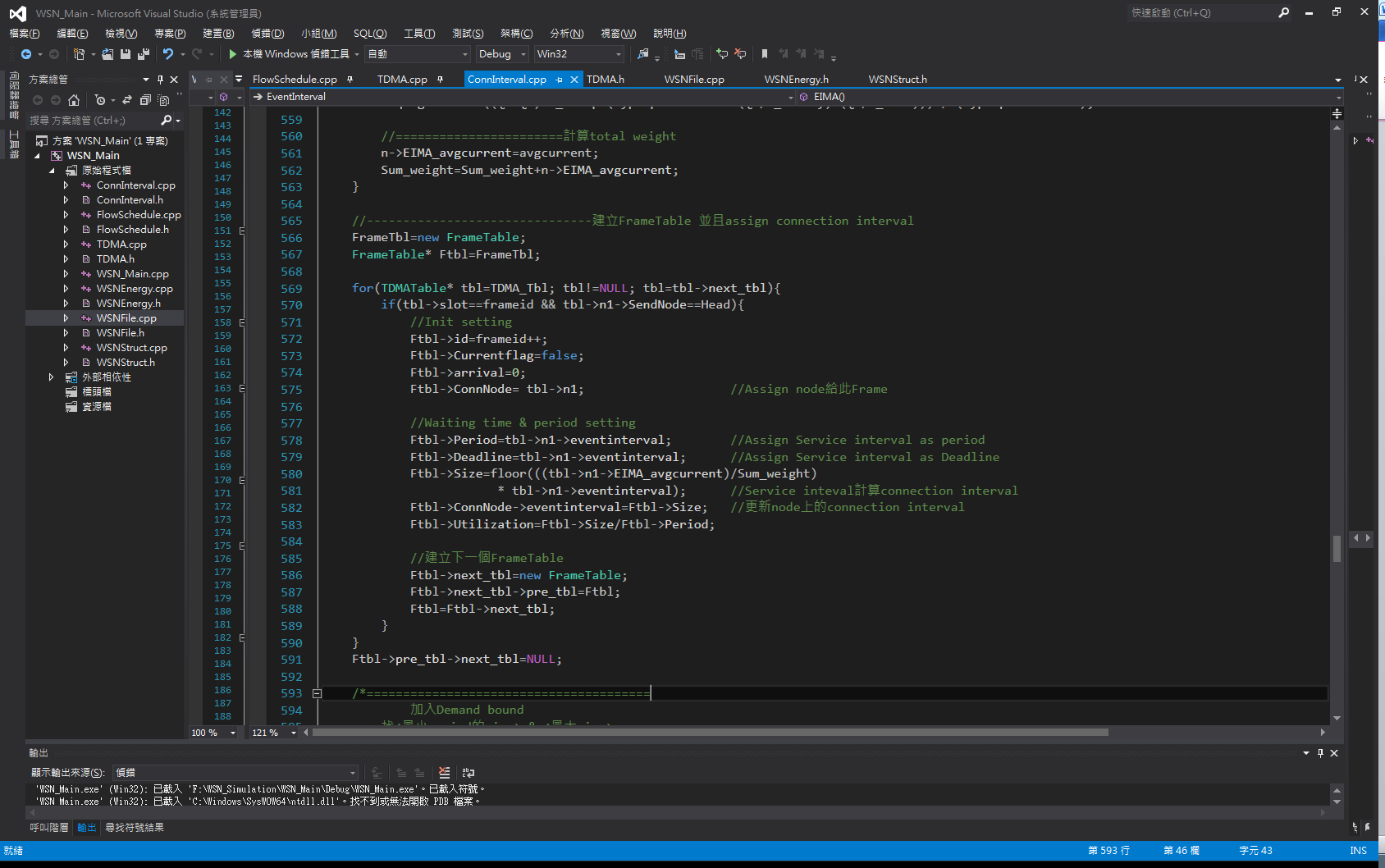
計算各個salve weight

Weight v1指的是不考量連傳性質(avgcurrent)

,MEI已考慮了

Weight v2指的是考量連傳性質(Seqavgcurrent)

Assign connection interval



FrameTable Setting

Period🡺Service Interval

Deadline🡺Service Interval

Size🡺Connection Interval

依照weight比例分配

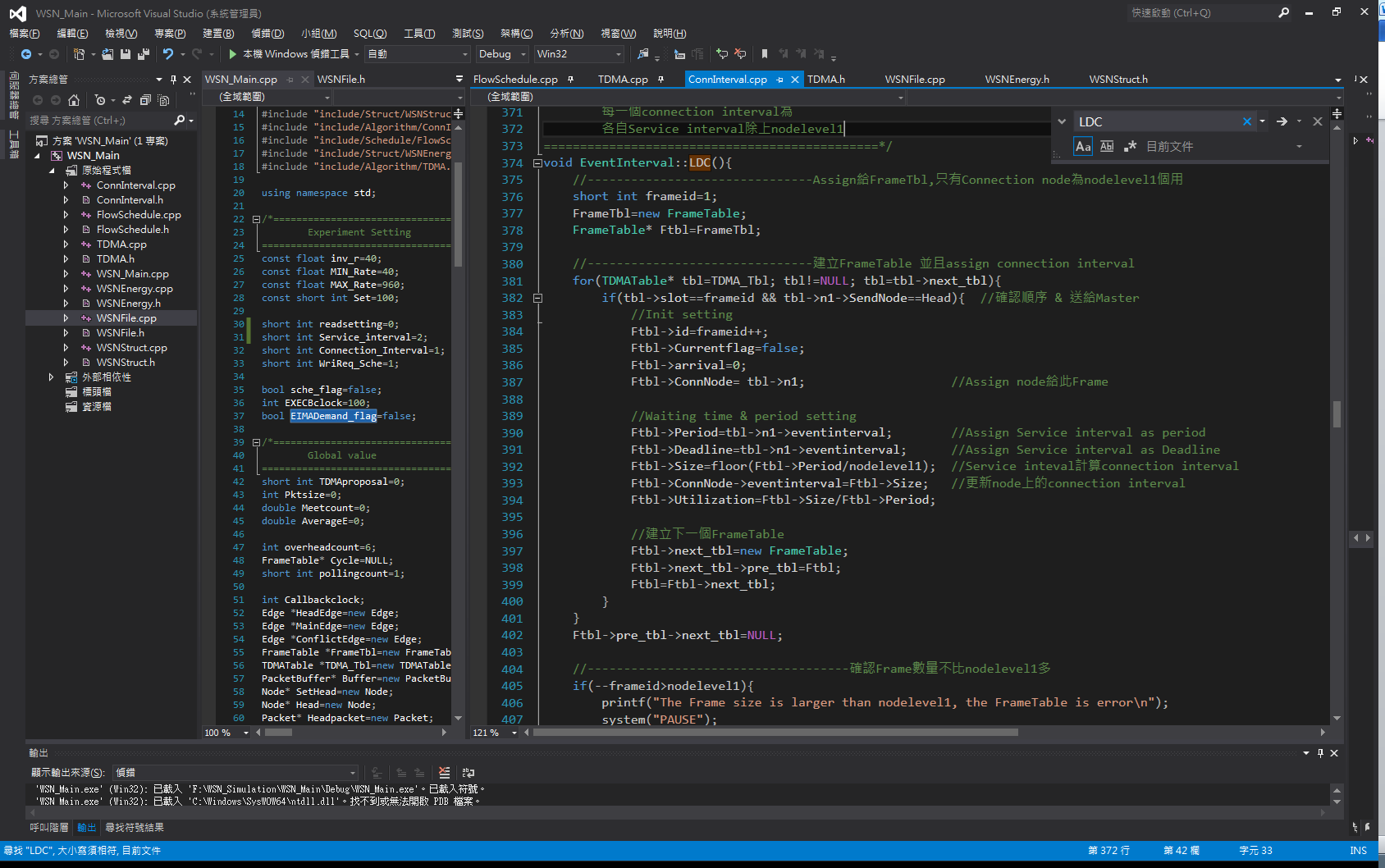
**(記得一定要在做ConnNode的connection interval修正)**

建立schedule的FrameTable

會依據TDMA slot的順序做assign

**🡺Low Duty Cycle (LDC)**

概念: 各個slave的connection interval依照**原本service interval**除上同level node數量，目的是要節省整體network耗能



FrameTable Setting

Period🡺Service Interval

Deadline🡺Service Interval

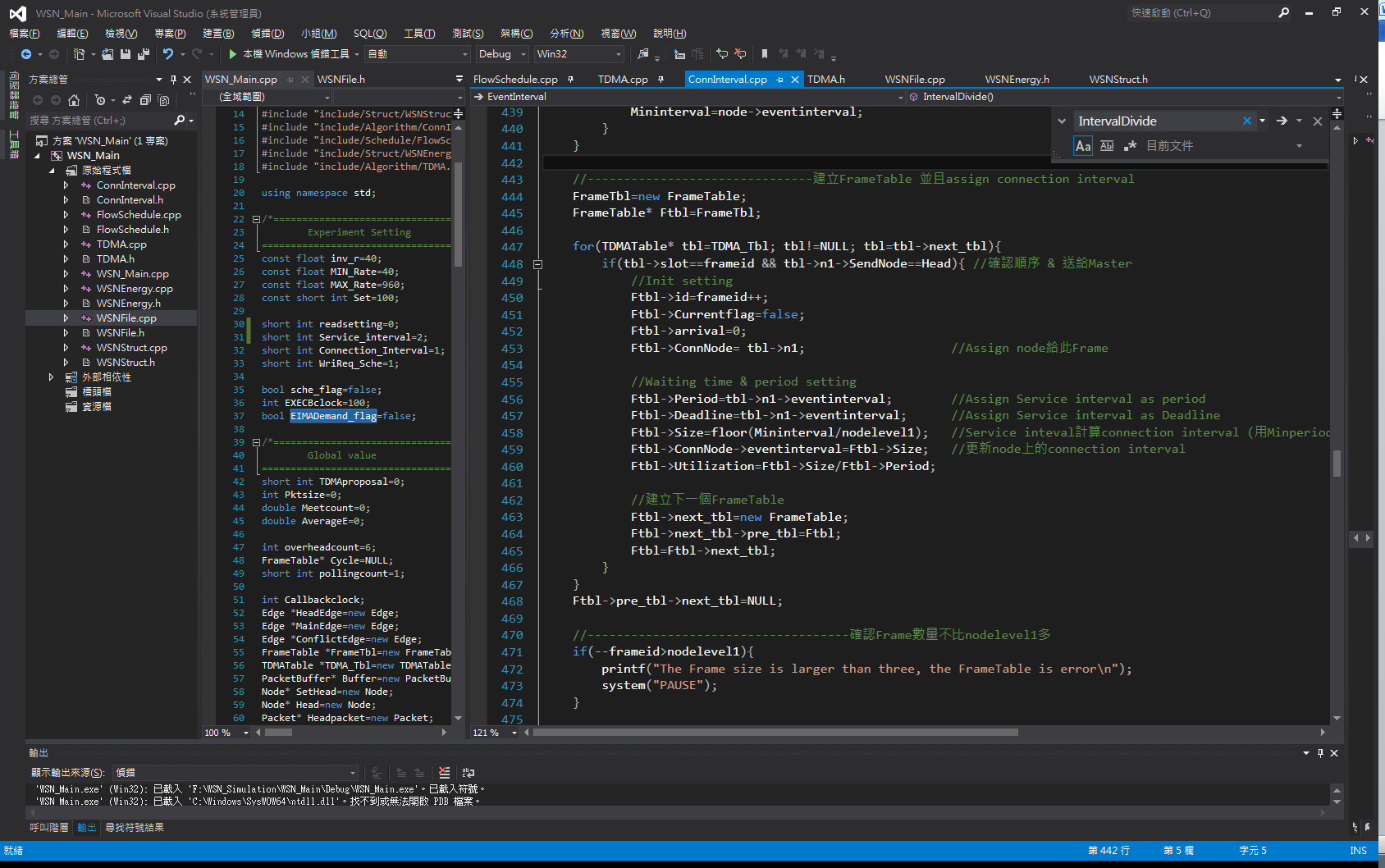
Size🡺Connection Interval

各個service interval除與同level node數量

**(記得一定要在做ConnNode的connection interval修正)**

**🡺IntervalDivide (Greedy)**

概念: 各個slave的connection interval依照**最小service interval**除上同level node數量，目的是要防止彼此間的block



FrameTable Setting

Period🡺Service Interval

Deadline🡺Service Interval

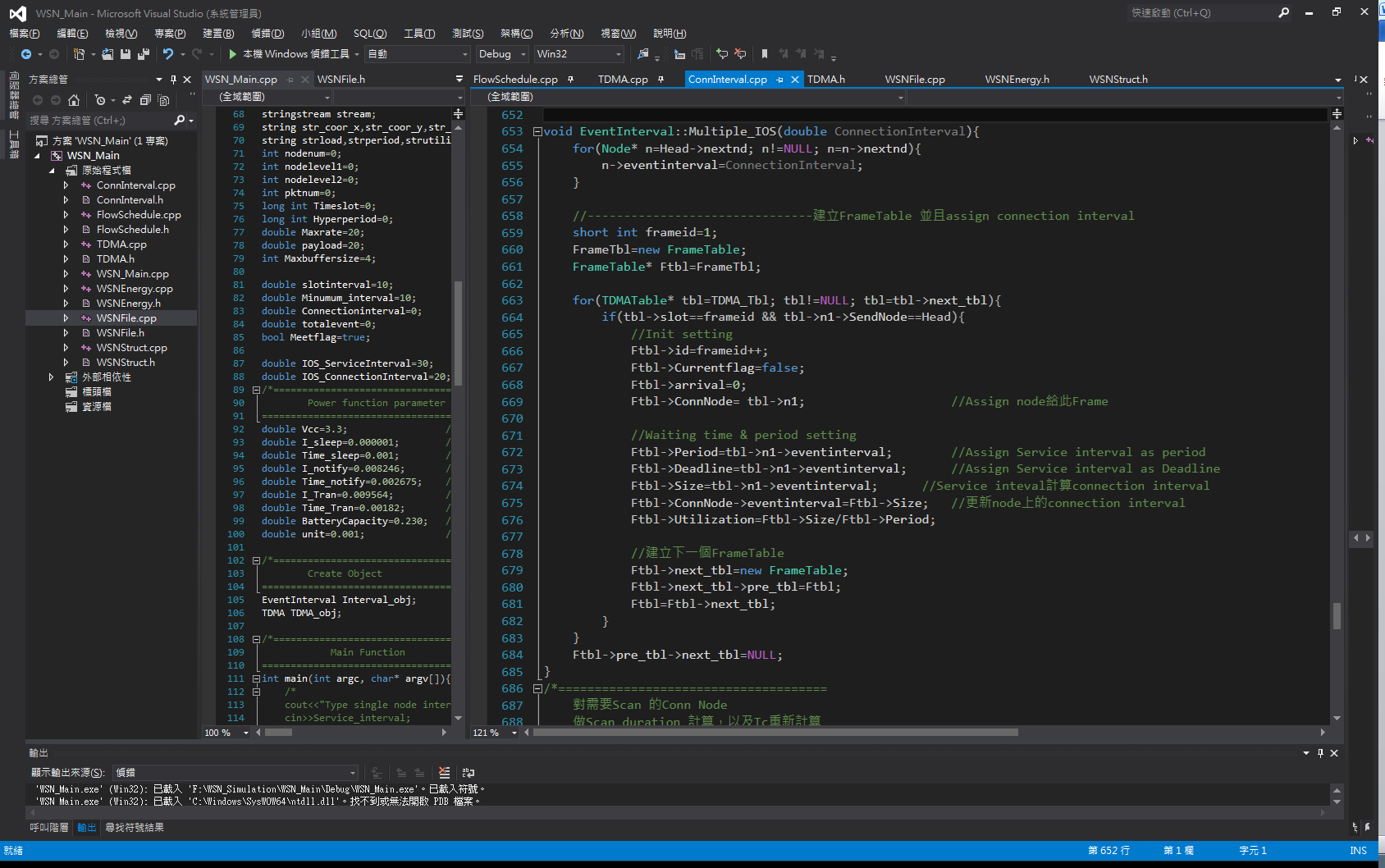
Size🡺Connection Interval

最小service interval除與同level node數量

**(記得一定要在做ConnNode的connection interval修正)**

**🡺Multiple\_IOS (IOS)**

概念: 20ms



FrameTable Setting

Period🡺 Connection Interval

Deadline🡺 Connection Interval

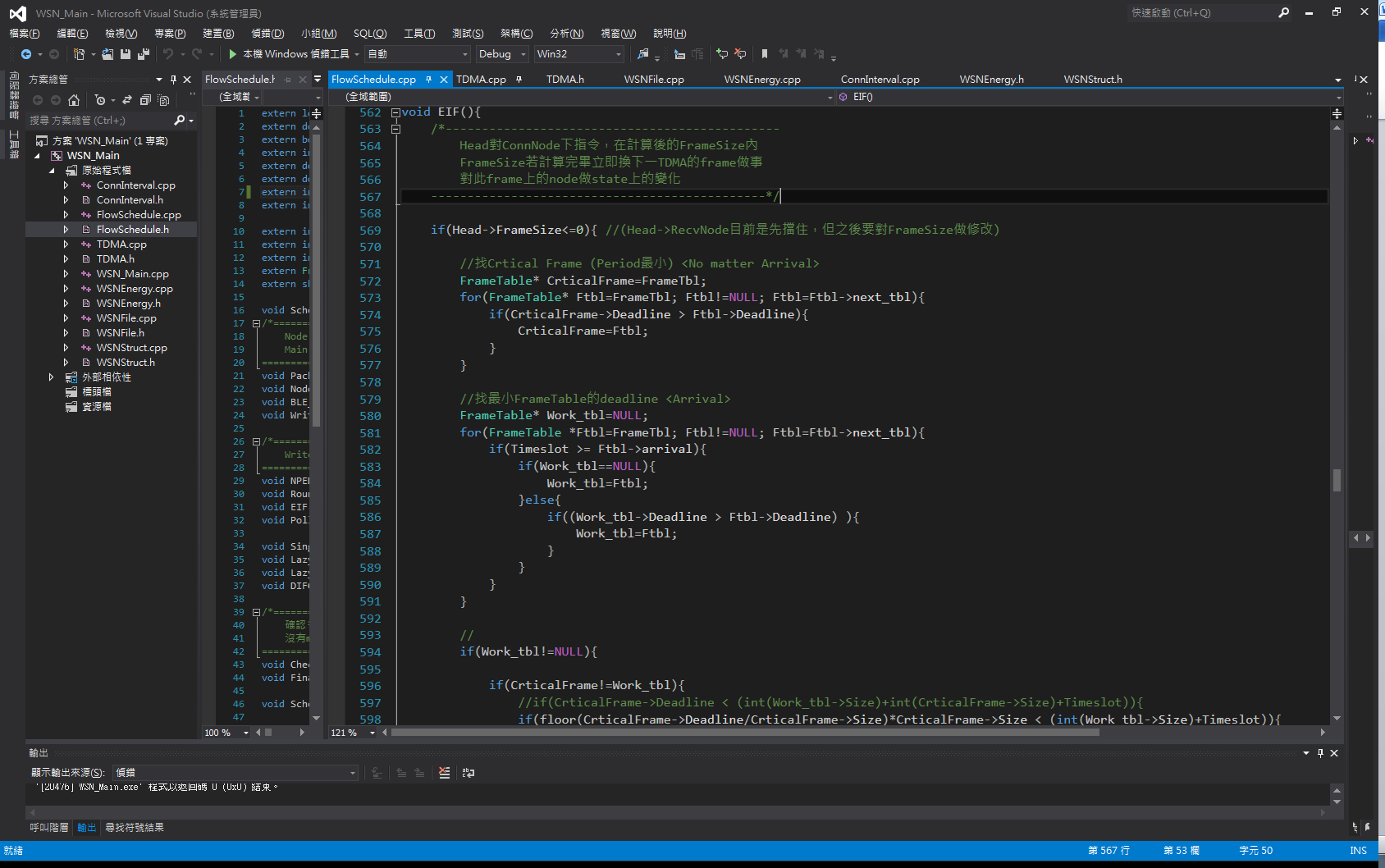
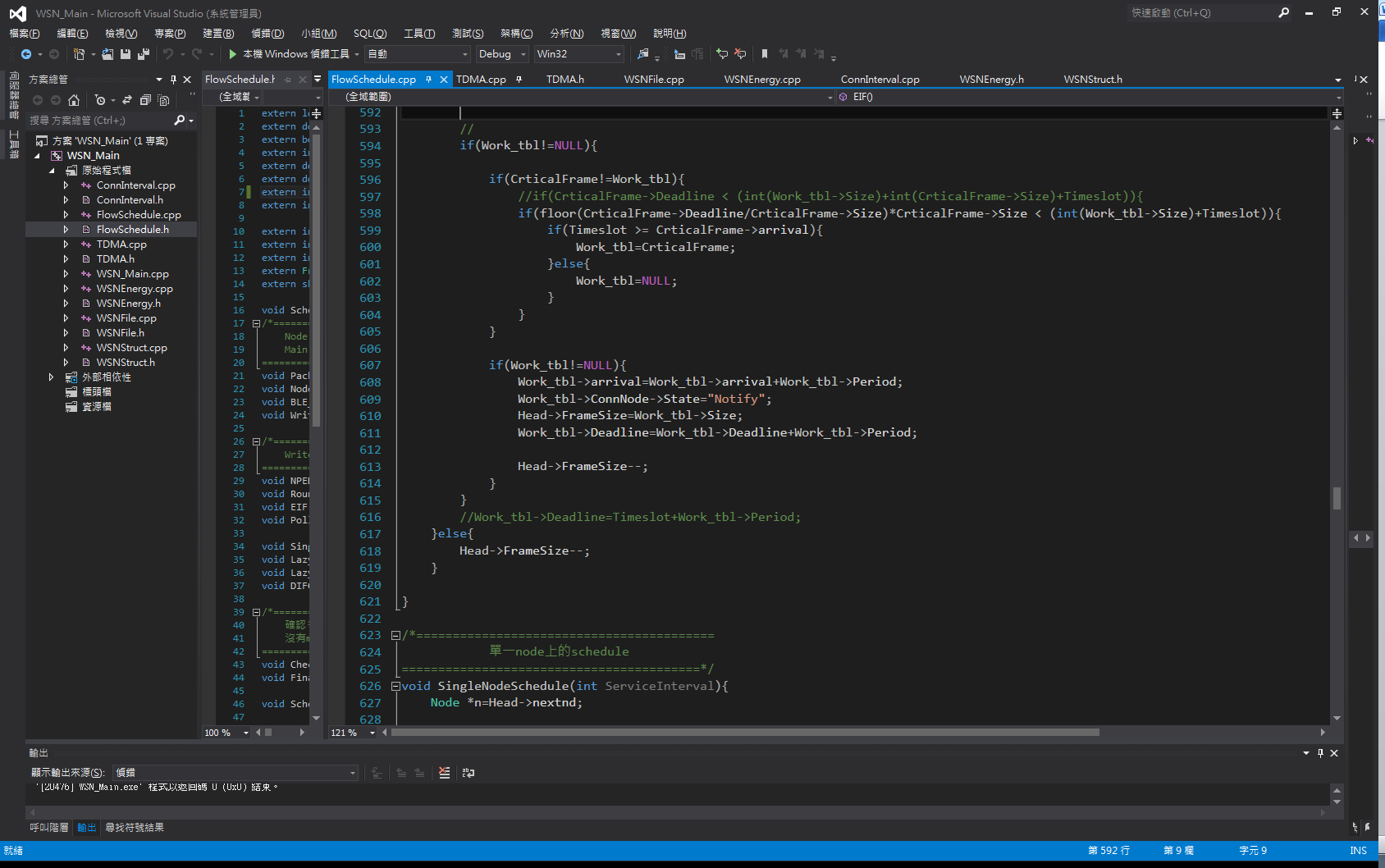
Size🡺Connection Interval

<IOS\_ConnectionInterval>

**(記得一定要在做ConnNode的connection interval修正)**

●Write-Request Scheduler

EIF



找最小deadline的Frame，不管是否arrival，解決block問題

CrticalFrame

做request

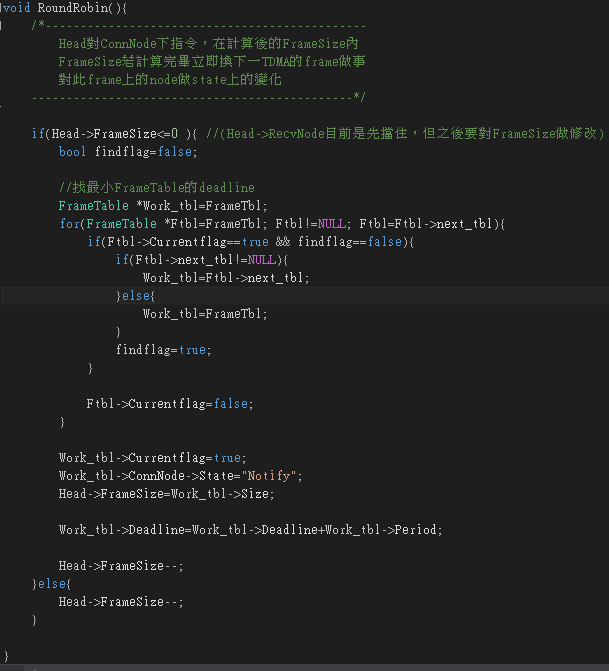
Node state會為Notify

做CrticalFrame和WorkTbl比較

找最小deadline的Frame

WorkTbl

Round Robin

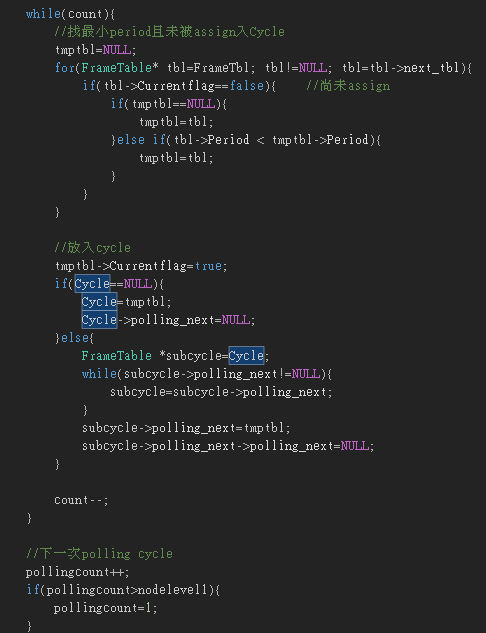


**連接node 發送request frameSize(slot)遞減**

**找deadline近的frame(slave)**

Polling(Lazy)

🡪與RR相似，但加入了優先權的概念，每個frame會從優先權高的開始搜尋{3,32,321}



**依據優先權大小放進cycle中**

Write\_Request

🡪依據前面的幾個schedule方法做request完之後，對node做傳輸判定

Notify State

>將State換成是Transmission

>RecvNode切換node

(因event時間為2.67ms所以Timeslot要加)

Energy計算

進行傳輸

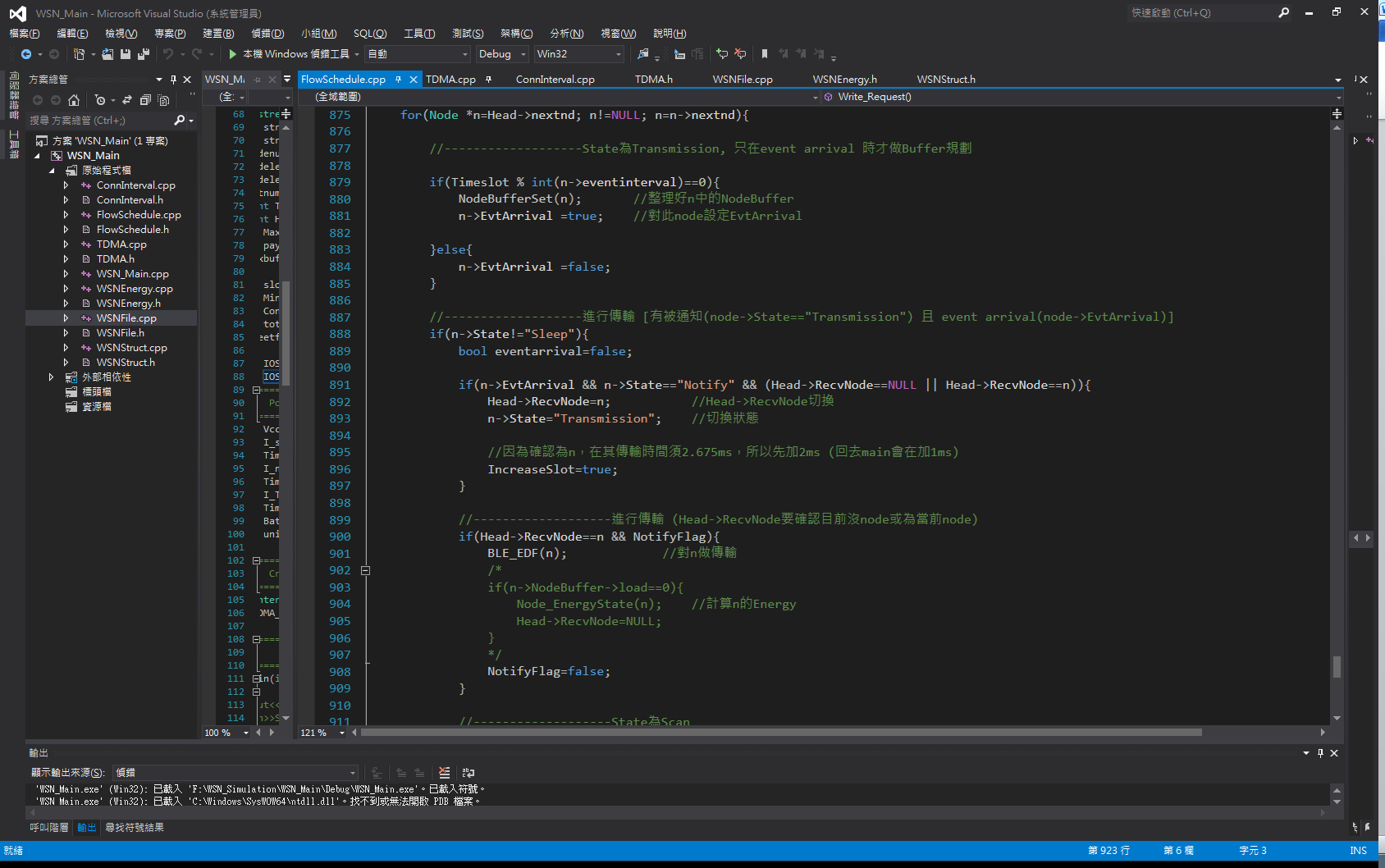
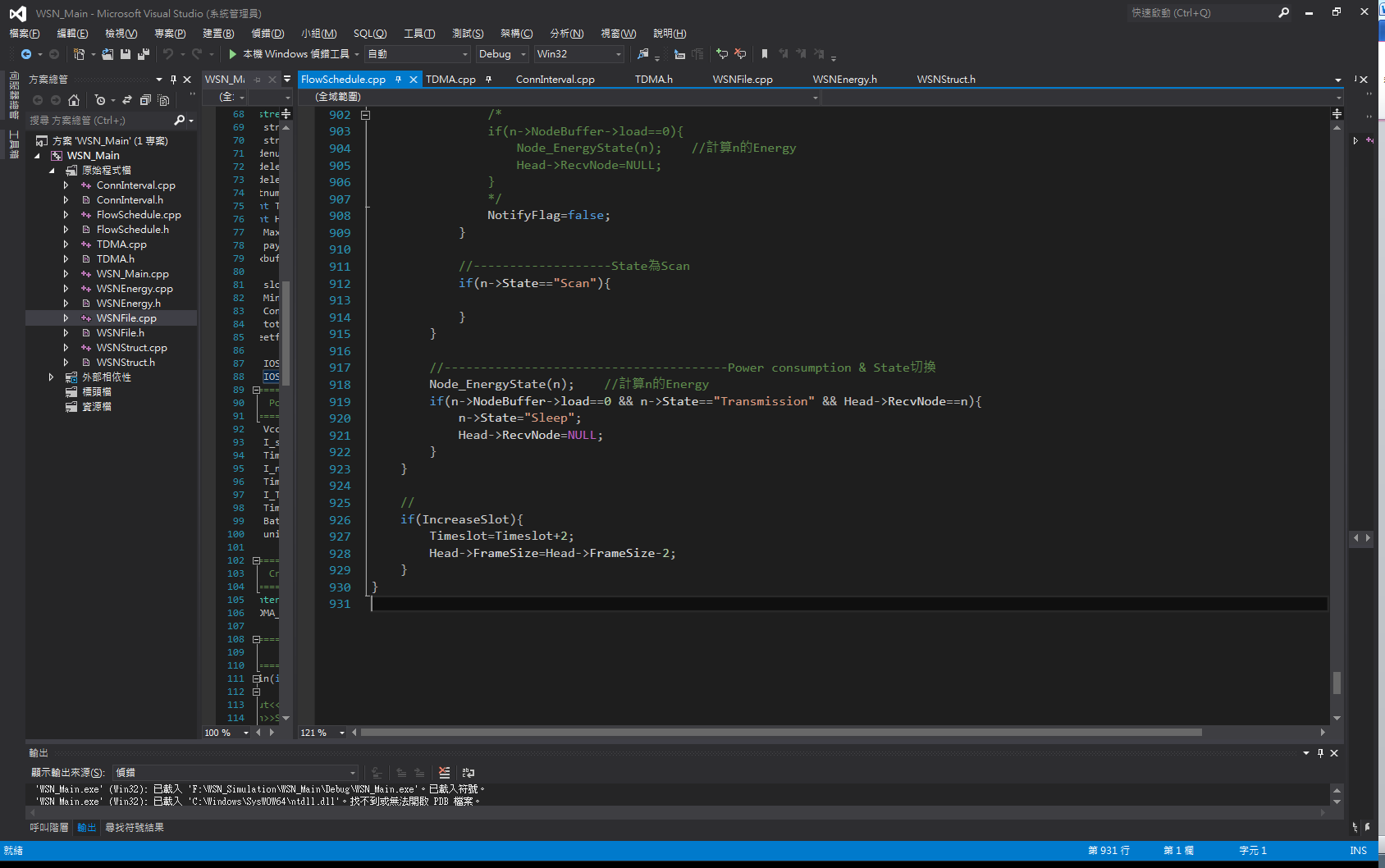
被通知或是連傳才會進入

(State為”Notify” or “Transmission”)

確認每一時間點下

Node是否會有connection event發生

(node的EvtArrival會trigger)



Write-Request中有三個function

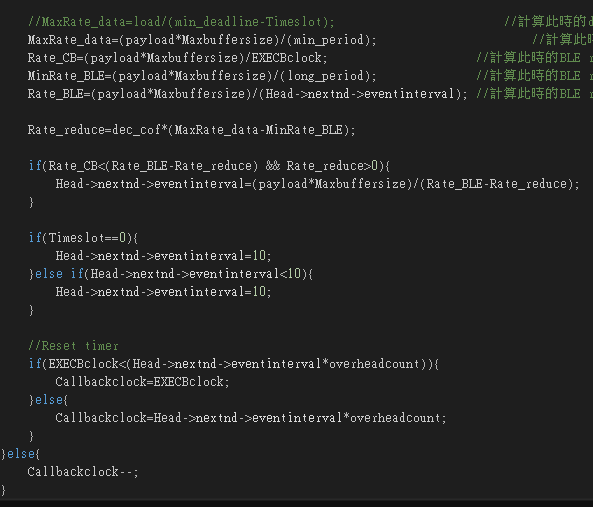
NodeBufferSet(Node) 🡺放入現在時間點下，priority最高的packet於Buffer中 (**pktsize**, **load**, and **pkt**)

BLE\_EDF(Node) 🡺對buffer做傳輸

Node\_EnergyState(Node)🡺計算event耗電量

●Lazy

計算目前運行的rate以及，當rate小於當前slave的最大rate時，會將運行的rate衝到最高，然後逐時遞減，當小於時又將其衝到最高。其中**遞減係數**以及**callback(sample)**的時間會相互影響到meet ratio及energy表現。



Slave app 最大rate

LazyCB的rate

Slave app 最小rate

當前運行的rate

遞減運行rate

遞減量=遞減係數\*(slave app rate range)

callback

●Write Output File

●Draw plt

**實作**

●檔案區分

避免路徑引起的問題要放C:\

基本功能在basic func setting.docx，mesh在mesh.docx

目前是1個event1個pkt，若要增加pkt，則在創建event那邊丟幾次buf，gateway端抓最後一個pkt就可以得到最後的執行時間。

判斷旗標在第9 byte，0xFF代表是最後一個封包，這會在notification那邊判斷event id來設定結尾。

Gateway端則在HCI解event裡加入RSPDict[20:22]判斷是否為0xFF才繼續下面判定meet與否。

※注意在gateway端是big endian排列，在IAR是little endian

●硬體設定

Keyfob debugger有點接觸不良多試幾次

●量測方法

10歐姆電阻量測

●idle time

SIF沒有blocking 但會有idle time

NPEDF 🡪 idle <= max(connection interval)

* Idle <= max( Cni, D - Cn )

D-C 顧慮高優先權被低的搶走，但高優先權可以有較高機會做，所以只算D-C

C 取最大的connection interval block別人的時間