Computer Network Spring 2010
Homework #3 Report

B96902120 施致誠
B96201044 黃上恩

2010/06/30

1 Compilation

輸入 make 或者

```
g++ main.cpp serv.cpp mysock.cpp log.cpp dv_algo.cpp -o DV_routing
```

2 Distance Vector Algorithm

每一台機器先初始化所有的距離，以及其 DV table。接下來跑一個 while 迴圈(main.cpp 20-35行)，每一次都先看看有沒有人丟訊息過來或是否有從stdin輸入指令 (呼叫 `Server::Wait()`，裡面呼叫了 `select()`，當 stdin 有東西時會回到 main，否則一直等待封包)，如果 `serv_sock` 所使用的 Socket 有任何動靜，就會呼叫 `Server::Refresh()`，更新 Distance Vector。每次更新 Distance Vector 之後，就會呼叫 `Server::DV_algo()`。在 `Server::DV_algo()`當中，每次都使用 Bellman-Ford Equation 重新計算該 Host 到所有其他 Host 之距離，即針對 i, j 跑兩層迴圈並計算

$$dis_x[i] = \min_j \{cost_x[j] + DV[j][i]\}$$

其中 j 跑遍所有 x 的鄰居。如果過程中 $dis_x[i]$ 有任何變更，計算完畢後，會重新呼叫 `Server::Send()` 將更新後的 Distance Vector 發送給所有的鄰居。

3 Solving Count-to-Infinity Problem

我們打算使用以下的方法：如果發現目前計算的距離變長了，就一口氣把值改成連往那個點的原本的 cost，再把 DV 傳給 neighbor。這個方法會比原本的快很多，考慮三個點 Server 0, Server 1, Server 2，以及他們之間的距離 $d(0,1) = 1, d(0,2) = 5, d(1,2) = 60$，用原本的方法將 $d(0,2)$ 更新爲 100 的時候，對 Server 0 來說需要 28 次的 Refresh，但是使用了新的方法，在 Server 0 上面僅執行了 2 次的 Refresh。